UNIVERSITY OF CALIFORNIA

Los Angeles

Deep 3D Embodied Visual Recognition

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Tong He

2021

ABSTRACT OF THE DISSERTATION

Deep 3D Embodied Visual Recognition

by

Tong He
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Stefano Soatto, Chair

Deep 3D embodied visual recognition refers to the scenarios where an autonomous agent needs to achieve some designated tasks (driving, searching, grasping, *etc*) in a 3D environment. To achieve the task at hand, the agent must have good geometric and semantic understandings of the 3D objects/scenes around it, such as the intention of a pedestrian and the pose/shape of an obstacle. In this thesis, I focus on driving-related problems. Methods for developing a self-driving agent can be roughly classified into two types: multi-stage and end-to-end models. 1) A typical multi-stage pipeline usually involves a series of steps such as perception, planning, and control. My works are mostly related to the perception module, which includes studying diverse 3D data representations (*e.g.*, 3D bounding boxes, morphable skeleton graphs, raw point clouds, parametric meshes, deep voxels, and implicit surface functions), and supervised/self-supervised 3D representation learning strategies. Different 3D representations and learning strategies have trade-offs among the requirements of the tasks, the availability of the data annotation and the modeling tools. 2) The other thread of my research is aligned with the end-to-end learned visual driving models, which utilize a single deep network to process the sensor data (*e.g.*, color images, LiDAR points) and estimate the low-level controls such as brake, throttle, and steering angle. The goal is to learn an autonomous agent that can generalize to various driving conditions, maps, and weather by imitation learning.

The dissertation of Tong He is approved.

Cho-Jui Hsieh

Guy Van den Broeck

Yizhou Sun

Stefano Soatto, Committee Chair

University of California, Los Angeles

2021

To my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I want to express my sincere thanks to my Ph.D. advisor, Prof. Stefano Soatto. He has been very encouraging and supportive of both my research and my well-being. I still remember that when I showed the manuscript of my first paper submission to Stefano, he told me that if a paper is not well-written and the idea is not well-explained then it should not be submitted at all. Basically, the original manuscript was written in a water-sink style, ignoring the underlying motivations, reasoning, and assumptions of the problem. Stefano guided me on how to formalize the problem as well as the method, and finally, the paper was in a good shape and got accepted as an oral AAAI paper. Last year, 2020, I was diagnosed with thyroid cancer and needed to have neck surgery. Stefano was being very considerate and caring during the whole time, keeping track of my health and mental conditions. He did his best to help me get through the surgery and I am recovering now.

I also want to thank my doctoral committee members: Prof. Yizhou Sun, Prof. Guy Van den Broeck and Prof. Cho-Jui Hsieh. Thank you for willing to serve on my doctoral committee and providing constructive advice on my advancement to candidacy talk as well as my Ph.D. final defense.

I want to thank all my colleagues and collaborators at UCLA Vision Lab: Konstantine Tsotsos, Jingming Dong, Nikolaos Kariannakis, Pratik Chaudhari, Alessandro Achille, Xiaohan Fei, Alex Wong, Safa Cicek, Xinzhu Bei, Shay Deutsch, Dong Lao, Peng Zhao, Alexandre Tiard, Albert Zhao, Stephanie Tsuei, Sim-lin Lau and all the others. Particularly, I want to express my thanks to Konstantine. He was my mentor when I first time stepped into the door of UCLA Vision Lab in 2015 summer. He led me into the field of SLAM and also helped build the bridge between me and UCLA.

I want to thank all my internship mentors: Haibin Huang, Hailin Jin, Yuanlu Xu, and Pei Sun. They offered me great opportunities to explore these cutting-edge research labs in the industry (*i.e.*, Megvii Research USA, Adobe Research, Facebook Reality Labs, Waymo

Research) and helped me develop my skills of collaboration and communication.

I want to thank all my close friends that I met at UCLA: Jieyu Zhao, Yitao Liang, Yujia Shen, Chi Zhang, Zhe Zeng, Chen Chen, Honghua Zhang, Tao Meng, Liunian Li, and Kuan-Hao Huang. I am grateful for all the love, friendships, companions, and supports from each one of you through my five wonderful Ph.D. years at UCLA. I want to thank Yitao especially. He is an amazing person and a great friend, *e.g.*, responsible, thoughtful, humorous, and smart. I have learned many good things from him and we have been continuously encouraging each other through our Ph.D. years.

Finally, I want to tell my mother: "I can not imagine how hard it is for you to raise me up alone. You are a great mother and a fabulous person. I love you".

VITA

2012–2016 B.S. in Computer Science, University of Electronic Science and Technology of China

2016–present Ph.D. student in Computer Science, University of California, Los Angeles

PUBLICATIONS

1. **Tong He***, Yuanlu Xu*, Shunsuke Saito, Stefano Soatto, Tony Tung. *Animation-Ready Clothed Human Reconstruction Revisited.* The International Conference on Computer Vision (ICCV), 2021. (equal contribution*)

2. Jialin Gao, **Tong He**, Xi Zhou, Shiming Ge. *Skeleton-Based Action Recognition With Focusing-Diffusion Graph Convolutional Networks.* IEEE Signal Processing Letters (SPL), 2021.

3. **Tong He**, John Collomosse, Hailin Jin, Stefano Soatto. *Geo-PIFu: Geometry and Pixel Aligned Implicit Functions for Single-view Human Reconstruction.* Conference on Neural Information Processing Systems (NeurIPS), 2020.

4. Albert Zhao*, **Tong He***, Yitao Liang, Haibin Huang, Guy Van den Broeck, Stefano Soatto. *SAM: Squeeze-and-Mimic Networks for Conditional Visual Driving Policy Learning.* Conference on Robot Learning (CoRL), 2020. (equal contribution*)

5. **Tong He**, John Collomosse, Hailin Jin, Stefano Soatto. *Enhancing the Fidelity of Novel View Synthesis from 3D Voxel Embeddings.* Asian Conference on Computer Vision (ACCV), 2020.

6. **Tong He**, Haibin Huang, Li Yi, Yuqian Zhou, Qihao Wu, Jue Wang, Stefano Soatto. *GeoNet: Deep Geodesic Networks for Point Cloud Analysis*. Conference on Computer Vision and Pattern Recognition (CVPR), 2019. (**5.6%, Oral Presentation**).

7. Yang Wang, Haibin Huang, Chuan Wang, **Tong He**, Jue Wang, Minh Hoai Nguyen. *GIF2Video: Color Dequantization and Temporal Interpolation of GIF images*. Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

8. **Tong He**, Stefano Soatto. *Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors*. AAAI Conference on Artificial Intelligence (AAAI), 2019. (**5.9%, Oral Presentation**).

9. **Tong He**, Yang Hu. *FashionNet: Personalized Outfit Recommendation with Deep Neural Network*. preprint arXiv: 1810.02443, 2018.

10. Zhihang Ren, **Tong He**, Shuaicheng Liu, Bing Zeng. *Shape Recovery of Endoscopic Videos by Shape from Shading using Mesh Regularization*. International Conference on Image and Graphics (ICIG), 2017. (**Oral Presentation**).

11. Heng Guo, **Tong He**, Shuaicheng Liu, Bing Zeng, Moncef Gabbouj. *Joint Video Stitching and Stabilization from Moving Cameras*. IEEE Transactions on Image Processing (TIP), 2016.

12. **Tong He**, Konstantine Tsotsos, Stefano Soatto. *Grid-based Uniform Feature Tracking for Visual-Inertial Navigation*. technical report: UCLA Cross-disciplinary Scholars in Science and Technology (CSST), 2015.

CHAPTER 1

Introduction

We must perceive in order to move, but we must also move in order to perceive.

- James Jerome Gibson, *The ecological approach to visual perception*, 1979

To navigate in the 3D world or interact with the surrounding objects, an autonomous agent must build some types of implicit or explicit 3D representations of the 3D objects and scenes around it. Such representations usually carry both geometric and semantic attributes (shapes, identities, affordances, *etc.*), and can inform the agent of the necessary knowledge that it needs to achieve the task at hand. Based on the 3D perception results, the agent should learn to proceed with its designated task in order to perceive more. An agent will get trapped in a dead state and never be able to achieve its goal if it does not have the ability to move. For example, a self-driving agent needs to maintain a positive speed in normal traffic situations and push the brake at red traffic lights. As explained above, deep 3D embodied visual recognition is a fundamental problem for any intelligent or autonomous agent. It involves two main problems: 1) building a task-dependent suitable 3D object/scene representation, 2) and learning how to move so as to achieve its goal.

Specifically, I focus on problems related to the driving scenarios. Depending on whether the aforementioned two problems are addressed separately or jointly, the methods of learning a self-driving agent can be roughly classified into two categories: multi-stage methods and end-to-end models. A typical multi-stage pipeline usually has a sequence of modules such as perception, planning, and control. Multi-stage pipelines are complicated engineering systems.

Each individual module could be further broken down into many research problems. For end-to-end models, all these different modules will be handled through a single deep neural network. The network inputs are data captured by the sensors, such as color images or LiDAR point clouds. The outputs are low-level control signals such as brake, throttle and steering angle. Currently, end-to-end models are less favored by commercial self-driving companies. Because compared with multi-stage systems, end-to-end models usually lack interpretability and system redundancy for safety protection. But end-to-end methods are expected to better handle complex driving situations that can not be effectively and explicitly modeled by manually designed rules. End-to-end models have recently demonstrated their usefulness on drones [LKR21], toy cars [CML18] and in driving simulators [DRC17], and thus this research direction is getting increasingly more attention from both academia and industry. Overall speaking, my Ph.D. research mainly focuses on the perception module of multi-stage systems, and the end-to-end learned models for autonomous driving agents.

1.1 3D Object and Scene Perception

Methods for 3D object and scene perception can be discussed from different angles, such as the trade-offs among diverse 3D data representations, or supervised and self-supervised 3D representation learning approaches. Here we will look into this research topic following the evolution of 3D data representations. Different representation methods should be determined based on the requirement of the task, the availability of the data and the modeling tools. At the early stage of 3D data based deep learning, some most often used representations include 3D bounding boxes [MAF17], 3D skeleton graphs [HS19] and spatially discretized voxels [WZX16]. These methods enable efficient representations of 3D object poses and shapes via commonly used computing kernels such as 2D or 3D convolutions and fully-connected layers [HZR16, PGM19, ABC16]. But they usually suffer from limited modeling capacities and large quantization errors when we are trying to encode objects with complex shapes and articulated

parts, such as furniture or human bodies. Later, with the advance of PointNet and Graph Neural Network [QSM17a, QYS17, DBV16], raw point clouds and (parametric) meshes are becoming increasingly more favored by 3D deep learning researchers. Although points and meshes are more flexible in modeling complex 3D shapes, they are still far from satisfaction towards the goal of representing sophisticated surface topology and are not suitable for unsupervised 3D representation learning. Acquiring high-quality and large-scale annotations for 3D objects and scenes is nontrivial, and therefore unsupervised 3D representation learning from multi-view images or video streams is a promising research direction. Recently, methods based on deep voxels [STH19a], deep implicit surface functions [PFS19a] and neural radiance field [MST20] have demonstrated unprecedented performance on many 3D computer vision tasks, such as object and scene reconstruction, detection and novel view rendering. My research works are tightly aligned with this evolution thread, covering diverse 3D data representations, and both supervised and self-supervised learning methods.

1.2 End-to-End Visual Driving Policy Learning

The goal of end-to-end visual driving policy learning is to learn an autonomous driving agent, that takes a color image and a turning signal as inputs, and directly estimates low-level control signals. We use an open-source photorealistic driving simulator, CARLA [DRC17], for agent training and evaluation. The simulator also contains other non-player agents such as vehicles and pedestrians, and can quantitatively measure various driving infractions like collision, driving out of the lane, breaking red traffic lights, and so on. After a driving model is trained, it will be deployed into the simulator. The agent will start at a source location and is expected to arrive at a specified destination. We evaluate its performance by measuring the route success rate. A route is successful if the agent can reach the target location within a certain time limit without any driving infractions. This is a challenging problem because the self-driving agent needs to perceive the environment, predict the intention of surrounding

agents, follow the traffic rules and keep moving towards its destination.

Our core method is based on behavior cloning, which is a type of imitation learning based driving policy learning method [CML18]. Basically, we can build a training dataset that consists of observation and action pairs collected from some expert agents. With such data, then we will be able to train an autonomous agent by supervised learning. This is just a vanilla version of the behavior cloning model. Researchers later found that we can leverage some side tasks to construct auxiliary losses in order to learn representations that can better generalize to different maps, weathers and driving conditions. For example, one of the most often used side tasks is semantic segmentation and the simplest way is having a two-stage driving model, where we first conduct semantic segmentation and then use the masks to learn a visual driving agent [MDG18, HXS19, BCP20, BKO19]. However, this approach usually suffers from propagation errors in estimating the side tasks. A more robust scheme than two-stage is multi-task learning, where the agent is supervised to achieve the main driving task and the side tasks jointly [XGY16, LMS18]. The latent embeddings of the learned agent conceptually contain two types of information: main-task relevant and irrelevant information. Ideally, we only want to learn the driving-relevant part. Because learning the main-task irrelevant information means that we are forcing the model to solve additional problems that are not useful for the core driving task.

Motivated by such analysis, we first train a privileged auto-driving agent that takes the ground-truth side task annotation as input and estimates the low-level controls. By doing so, the latent representations of the agent network are expected to encode squeezed main-task relevant information from the side tasks. As for information that is irrelevant to the driving task, it is expected to be excluded in these learned latent embeddings. We call this model the squeeze network. Additionally, we will train a mimic network which contains the self-driving agent that will be used at inference time. Its inputs only contain observations that are available during inference, not including any ground-truth side task annotation. Besides the control estimation losses, we also apply a latent embedding regularizer from the squeeze

network to the mimic network. Therefore, the learned representations of the mimic network can be regularized by the driving relevant information squeezed from the side tasks. Note that while two-stage and multi-task pipelines are both trying to estimate the side tasks, our squeeze-and-mimic method does not need to do so. We directly use the ground-truth side task annotation as the input, and thus our model is free from side task estimation errors.

1.3 Organization of the Thesis

Chapter 2. We first study an ill-posed problem of monocular 3D vehicle detection [HS19]. The main idea is optimizing two-scale (*i.e.*, fine and coarse scales) projection consistency between the generated 3D hypotheses and their 2D pseudo-measurements. Specifically, we use a morphable 3D skeleton graph to generate a fine-scaled representation of vehicle shape and pose. To reduce its sensitivity to 2D landmarks, we jointly model the 3D bounding box as a coarse representation which improves robustness. We also integrate three task priors, including unsupervised monocular depth, a ground plane constraint and vehicle shape priors, with forward projection errors into an overall energy function. Our method shows state-of-the-art monocular 3D vehicle detection performance on the KITTI benchmark [GLU12].

Chapter 3. When representing objects with sophisticated shapes, 3D bounding boxes and skeletons will suffer from limited modeling capacities and large approximation errors. In contrast, meshes and point clouds can faithfully encode object surfaces. Our basic observation is that surface-based geodesic topology provides strong cues for object semantic analysis and geometric modeling. However, such mesh-based topology information is lost in point clouds. Thus we introduce GeoNet [HHY19], a deep neural network trained to model the intrinsic structure of surfaces represented as point clouds. We demonstrate that the learned geodesic-aware latent features can capture the underlying surface topology of a point cloud, and further benefit multiple point set analysis tasks, including point upsampling, normal estimation, mesh reconstruction and non-rigid shape classification.

Chapter 4 and 5. Since meshes and point clouds are usually sparsely distributed object surface samples, it is difficult to reconstruct a complete 3D shape. Meanwhile, deep implicit surface functions are based on dense occupancy fields, with which a watertight mesh of an object can be directly reconstructed by the Marching Cubes algorithm. It motivates us to propose hybrid 3D representations [HCJ20b, HXS21] that exploit the shape reconstruction flexibility of deep implicit functions, and the topology modeling regularity of deep voxels and parametric meshes. Particularly, we extract pixel-aligned appearance features from the high-resolution input images and geometry-aligned spatial features from the underlying coarse shape estimation. These features are passed into MLP-based deep implicit surface functions for learning dense 3D occupancy fields. Our reconstructions achieve high quality and realism across different views, *e.g.*, both the visible and the occluded sides. We further propose to decorrelate object poses in a canonical space, whose reconstructions are animatable by a motion sequence and useful for downstream tasks such as VR/AR and robotic simulation.

Chapter 6. Besides learning supervised 3D representations, we also study self-supervised methods leveraging large-scale unannotated videos/images. We present a novel view synthesis method [HCJ20a] based upon latent voxel embeddings of an object, which encode both shape and appearance information and are learned without explicit 3D occupancy supervision. Our method uses an encoder-decoder architecture to learn such deep volumetric representations from multi-view images. The core idea is lifting the input image features from the 2D plane into a predefined 3D volume (*i.e.*, voxel-shape autoencoder bottleneck), and then projecting the latent voxel embeddings to a target camera pose and rendering a novel view. By enforcing reconstruction losses on the predicted images, we can backpropagate gradients through this encoder-decoder network and achieve self-supervised learning of the latent voxel features, which could be used for novel-view prediction, 3D detection, robotic navigation, and *etc.*

Chapter 7. For end-to-end learned visual driving models, we describe a policy learning approach [ZHL19] to map visual inputs to low-level controls conditioned on turning command.

Our model leverages side tasks on semantics and object affordances via a learned representation trained for driving. To learn this representation, we train a squeeze network to drive using annotations for the side task as input. This representation encodes the driving-relevant information associated with the side task while ideally throwing out side task-relevant but driving-irrelevant nuisances. We then train a mimic network to drive using only images as input and use the squeeze network's latent representation to supervise the mimic network via a mimicking loss. Notably, we do not aim to achieve the side task nor to learn features for it; instead, we aim to learn, via the mimicking loss, a representation of the side task annotations directly useful for driving. We test our learned self-driving agent in a photorealistic simulator and demonstrate that it can generalize to various driving scenarios, maps and weathers.

CHAPTER 2

Monocular 3D Vehicle Detection

2.1 Introduction

Objects are regions of three-dimensional (3D) space that can move independently as a whole and have both geometric and semantic attributes (shapes, identities, affordances, etc.) in the context of a task. In this paper, we focus on vehicle objects in driving scenarios. Given an image, we wish to produce a posterior probability of vehicle attributes in 3D, or at least some point-estimates from it.

Inferring 3D vehicles from a single image is an ill-posed problem since object attributes exist in 3D but single images can only provide partial pseudo-measurements in 2D. Therefore, we propose to solve this task by tackling two issues: (i) how to ensure 3D-2D consistency between the generated 3D vehicle hypotheses and their corresponding 2D pseudo-measurements, which requires strong 3D hypotheses generators as well as robust scoring mechanisms; (ii) how to refine 3D hypotheses with task priors that can be integrated into an easy-to-optimize loss function.

For the first problem, we use a joint modeling method that leverages two different 3D hypotheses generation schemes: one serves as a coarse representation of vehicle shape and pose while the other is fine-scaled. We design end-to-end trained deep networks to generate 3D hypotheses for each vehicle instance in the form of both 3D bounding box and morphable wireframe model (a.k.a linear shape model). Shape and pose parameters will be adjusted according to the 2D pseudo-measurements via an optimization approach. A wireframe

model can determine shape and pose more precisely than a 3D bounding box, but it is very sensitive to the 2D landmark measurements which can be easily affected by issues like partial occlusions, shadow, low resolution, etc. Therefore, we jointly model the 3D bounding box projection constraint to improve its robustness. We conduct ablation studies to demonstrate benefits brought by jointly modeling the coarse and the fine-scaled 3D object pose and shape representations.

For the second problem, we consider three constraints on vehicles. Cars should stay on the ground plane, should look like a car, and should be at a reasonable distance from the observation camera. The first argument serves as a supporting plane constraint for vehicles. The second argument is a prior term for vehicle shapes. The last argument indicates that vehicle translation in camera coordinates should be constrained by a monocular range map of the current driving scene. These constraints are jointly modeled with 3D-2D consistency terms in order to further improve vehicle shape and pose estimation.

In summary, in this paper we propose an approach for vehicle 3D shape and pose estimation from a single image that leverages the coarse and the fine-scaled 3D hypotheses, as well as multiple task priors, as shown in Fig. 2.2, for joint inference. Our contributions are:

- We propose and empirically validate the joint modeling of vehicles' coarse and fine-scaled shape and pose representations via two complementary 3D hypotheses generators, namely 3D bounding box and morphable wireframe model.
- In our method, we model multiple priors of the 3D vehicle detection task into easy-to-optimize loss functions. Such priors include unsupervised monocular depth, a ground plane constraint and vehicle shape priors.
- We build an overall energy function that integrates the proposed two improvements which improves the state-of-the-art monocular 3D vehicle detectors on the KITTI dataset.

Figure 2.1: Representative 3D detection results, shown as projections on the input images. 3D morphable shape models of each vehicle are colored in green. Dashed green lines are occluded edges. For the 14 vertices of each morphable shape model, visible vertices are colored in red and occluded ones in yellow. 2D bounding boxes are colored in blue.

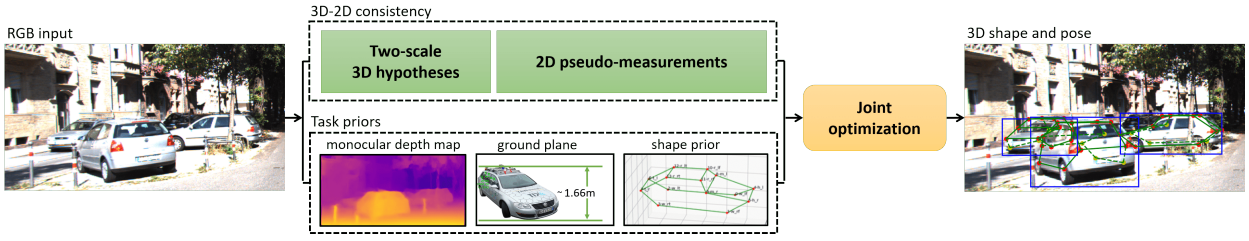


Figure 2.2: Our system takes a single image as input, and generates vehicles' 3D shape and pose estimation in camera coordinates.

2.2 Related Work

To produce rich descriptions of vehicles in 3D space, methods leveraging various data are proposed, such as video, RGB-D, or RGB, among which RGB methods are most related to our work.

Video methods: In [SC15, MSK17, CRU16, DFS17, FS18], temporal information is explored for (moving) 3D objects localization by a recursive Bayesian inference scheme or optimizing loss functions that are based on non-rigid structure-from-motion methods [THB03].

RGB-D methods: MV3D [CMW17] encodes lidar point clouds into multi-view feature maps, which are fused with images, and uses 2D convolutions for 3D localization. In contrast, F-PointNet [QLW18] directly processes lidar point clouds in 3D space using two variants of PointNet [QYS17] for 3D object segmentation and amodal detection. Other methods that also use lidar point clouds include [RS18, XAJ18]. 3DOP [CKZ15] exploits stereo point clouds and evaluates 3D proposals via depth-based potentials.

RGB methods: Mono3D [CKZ16] scores 3D bounding boxes generated from monocular images, using a ground plane prior and 2D cues such as segmentation masks. Deep3DBox [MAF17] recovers 3D pose by minimizing the reprojection error between the 3D box and the detected 2D bounding box of the vehicle. Task priors are not jointly modeled with 3D-2D innovation terms. 3DVP [XCL15] proposes 3D voxel with occlusion patterns and uses a set of ACF detector for 2D detection and 3D pose estimation. Its follow-up work, SubCNN [XCL17], uses deep networks to replace the ACF detectors for view-point dependent subcategory classification. Active shape models are explored in [ZSS11, ZSS13, ZSS14a, ZSS14b] for vehicle modeling. CAD models are rendered in [MXS15, CSC15] for 3D detection by hypotheses sampling/test approaches using image features, such as HOG [DT05]. In the state-of-the-art DeepMANTA [CCR17], vehicle pose is adjusted by 3D-2D landmark matching. These approaches only model either the coarse or the fine-scaled 3D shape and pose representation of a vehicle, thus have limitations in accuracy and robustness. Moreover, task priors, such

as monocular depth of the current driving scene, vehicle shape priors, supporting plane constraints, etc., are only partly considered and not jointly optimized with forward projection errors.

2.3 Methodology

We wish to infer the posterior distribution of object pose $g \in SE(3)$ and shape $S \subset \mathbb{R}^3$, given an image I, $P(g, S|I)$, where $SE(3)$ denotes the Euclidean group of rigid motions that characterize the position and orientation of the vehicle relative to the camera, and shape S is characterized parametrically for instance using a point cloud or a linear morphable model. In Sec. 2.7 we describe all the modeling assumptions needed to arrive at a tractable approximation of the posterior, maximizing which is equivalent to minimizing the weighted sum:

$$E(g, S) = E_{2D3D} + \lambda_1 E_{LP} + \lambda_2 E_{MD} + \lambda_3 E_{GP} + \lambda_4 E_S \tag{2.1}$$

in which the first two terms indicate forward projection errors of the coarse and the fine-scaled 3D hypotheses. The last three terms, respectively, represent constraints enforced via unsupervised monocular depth, a ground plane assumption, and vehicle shape priors. In the next few sections we formalize and introduce details of our inference scheme, as reflected in the joint energy function Eq. (2.1), and how we generate the 3D hypotheses as well as the 2D pseudo-measurements via deep networks to facilitate its optimization.

2.3.1 Notation

We assume we are given a color image $I : D \subset \mathbb{R}^2 \to \mathbb{S}^2$ sampled as a positive-valued matrix. An axis-aligned subset $b \subset D$ is called "2D bounding box", and represented by the location of its center $(t_x, t_y) \in \mathbb{R}^2$, and scales $(e^w, e^h) \in \mathbb{R}^2_+$, all in pixel units and represented in exponential coordinates (w, h) to ensure positivity. We assume the camera is calibrated so these can be converted to Euclidean coordinates. Equivalently, a 2D bounding box can be

represented by an element of the scaled translation subgroup of the affine group in 2D:

$$g_b = \begin{bmatrix} e^w & & t_x \\ & e^h & t_y \\ & & 1 \end{bmatrix} \in \mathbb{A}(2). \tag{2.2}$$

In space, we call a gravity-aligned parallelepiped $B \subset \mathbb{R}^3$ a 3D bounding box, resting on the ground plane, whose shape is represented by three scales $\sigma = (e^L, e^H, e^W) \in \mathbb{R}^3_+$, again in exponential coordinates to ensure positivity, and whose pose is represented by its orientation $\theta \in [0, 2\pi)$ and position on the ground plane $T \in \mathbb{R}^3$ relative to the camera reference frame. A 3D bounding box can also be represented as an element of the scaled translation subgroup of the affine group in 3D:

$$g_B = [R(\theta),\ e_4] \begin{bmatrix} e^L & & & T_X \\ & e^H & & T_Y \\ & & e^W & T_Z \\ & & & 1 \end{bmatrix} \in \mathbb{A}(3) \tag{2.3}$$

where $R(\theta)$ is a rotation around Y by θ, so $(R(\theta), T) \in SE(3)$, and $e_4^T = [0,\ 0,\ 0,\ 1]$. Here $T_Y = 0$ is the camera height from the ground plane. Assuming the ground plane is represented by its (scaled) normal vector $N \in \mathbb{R}^3$, we describe it as the locus $\{T \in \mathbb{R}^3 \mid N^T T = 1\}$ (the ground plane cannot go through the optical center). Therefore, the vector T is subject to the constraint $N^T T = 1$.

We call $S \subset \mathbb{R}^3$ a shape, represented by a set of K points[1] $P_k \in \mathbb{R}^3$ in a normalized reference frame. Note that $p \in D \subset \mathbb{R}^2$ are corresponding landmark points within the 2D bounding box. Equivalently, $S \in \mathbb{R}^{3\times K}/\mathbb{A}(3)$ is in the affine shape space [Ken84] of K points, where the quotient is restricted to transformations of the form Eq. (2.3). $Z : D \subset \mathbb{R}^2 \to \mathbb{R}_+$ is a depth map, that associates to each pixel $(x, y) \in D$ a positive scalar $Z(x, y)$.

[1]We overload the notation and use P for for points in space and probabilities. Which is which should be clear from the context.

2.3.2 Inference scheme

As shown in Fig. 2.2, the inference criterion we use combines a generative component, whereby we jointly optimize the innovation (forward prediction error) between the projection of the 3D hypotheses and the image pseudo-measurements, monocular depth map constraints, geometric constraints (ground plane), in addition to penalizing large deformations of the shape prior. In Sec. 2.7 we derive an approximation of the posterior of 3D pose and shape $P(g_B, S|I)$, maximizing which is equivalent to minimizing the negative log:

$$-\log[P(g_b|g_B)P(p|g_B,S)P(Z_b|T_Z)P(T)P(S)] \tag{2.4}$$

which is in accordance with Eq. (2.1). The first term is the 2D bounding box compatibility with the projected 3D bounding box. It is a coarse statistic of the geometric innovation.

$$E_{2D3D} = \|g_b - \pi(g_B)\| \tag{2.5}$$

where the subscript suggests 2D/3D consistency, and π denotes the central perspective map, assuming a calibrated camera. The second term is the fine-scaled geometric innovation, *i.e.*, the distance between the predicted position of projected wireframe model vertices, and their 2D pseudo-measurements by a landmark detection network.

$$E_{LP} = \sum_{k=1}^{K} \|p_k - \pi(g_B P_k)\|_2^2 \tag{2.6}$$

where LP stands for landmark projection. To approximate the third term, we produce a dense range map of the current driving scene via an unsupervised monocular depth map estimation network.

$$E_{MD} = \|T_Z - Z_b\| \tag{2.7}$$

where MD means monocular depth, and $Z_b \in \mathbb{R}$ is the average depth of an image crop specified by $(I, g_b) : D \subset \mathbb{R}^2 \to \mathbb{S}^2$. The fourth term assumes a geometric constraint by the ground plane, characterized by the normal vector N.

$$E_{GP} = \|N^T T - 1\| \tag{2.8}$$

in which GP indicates ground plane. As generic regularizers, we also assume small deformations (shape coefficients α_n close to their mean) of the shape model.

$$E_S = \sum_{n=1}^{N} \|\alpha_n - \frac{1}{N}\sum_n \alpha_n\|_2^2 \tag{2.9}$$

The overall loss function is the weighted sum, with multipliers λ:

$$E = E_{2D3D}(g_b, g_B) + \lambda_1 E_{LP}(p, g_B, P) + \lambda_2 E_{MD}(Z_b, T) + \lambda_3 E_{GP}(T) + \lambda_4 E_S(\alpha) \tag{2.10}$$

2.3.3 Mono3D++ network

Our inference scheme leverages independence assumptions to factor the posterior probability into components, resulting in the compound loss described above. To initialize the minimization of Eq. (2.10), we separate it into modules that are implemented as deep networks, which is shown in Fig. 2.3. In the next paragraphs we describe each component in more details.

2D Bounding box. We use a one-stage detection architecture similar to SSD [LAE16] to estimate a posterior over (a regular subsampling of) the set of bounding boxes, $P(l, g_b|I)$. Before setting putative bounding boxes on latent feature maps at different scales, we fuse feature maps from shallow and deep layers. It is shown in [RCL17] that this step can improve both the semantic meaning and the resolution of the feature maps for better object detection.

$$H_{P,Q}(l_j, \hat{l}_i) + \lambda d(g_{b_j}, \hat{g}_{b_i})\chi(l_j) \tag{2.11}$$

where $j = j(i) = \arg\max_j IoU(b_j, \hat{b}_i)$. Here H denotes the cross-entropy between the true class posterior $P(l|I)$ and the model realized by our network $Q(\hat{l}|I)$, and d is a distance in $\mathbb{A}(2)$ that sums the Euclidean distance of the translational terms and the exponential coordinates of the scale terms, (w, h). Here i is the index of each putative bounding box (a.k.a. "anchor box") which is sampled with negative/positive sample ratio of 3:1 from a larger group of regularly sampled anchor boxes on multiple latent feature maps. The index j to be chosen to match i consists of a data association problem [BAD17]. We apply an

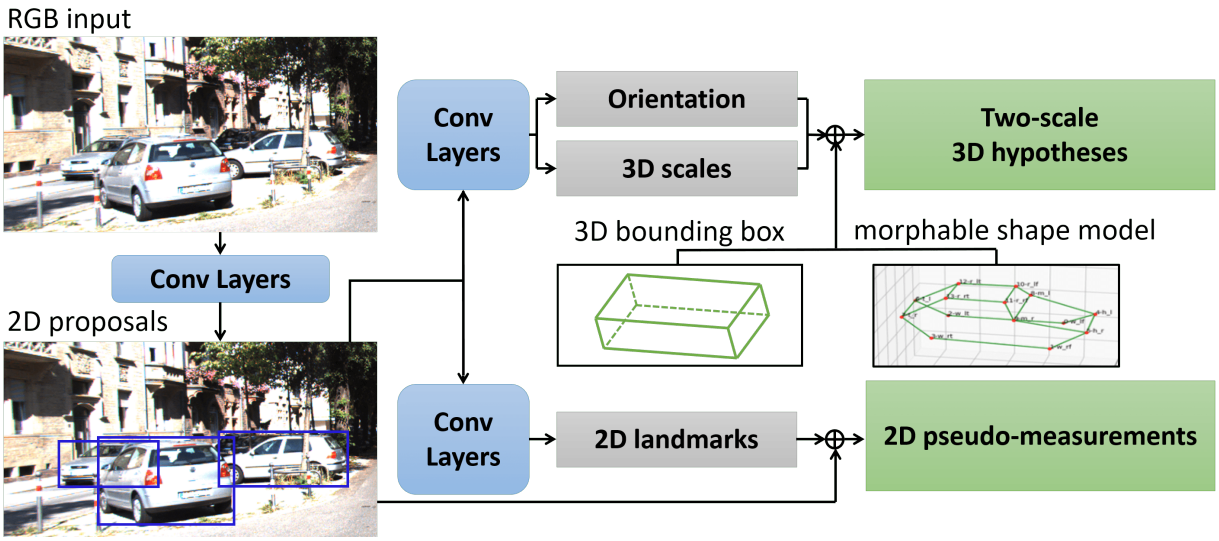


Figure 2.3: The two-scale 3D hypotheses consist of the rotated and scaled 3D bounding box and morphable wireframe model. The image pseudo-measurements include 2D bounding boxes and landmarks. In our inference scheme, we use the hypotheses and the pseudo-measurements to initialize the optimization of Eq. (2.10) and generate the final 3D pose and shape estimation of a vehicle.

indicator function $\chi(\cdot)$ before the bounding box coordinate regression term so that this loss is only optimized for positive anchor boxes.

2D Landmark. We employ a stacked hourglass network [NYD16], with skip-connections, to approximate the posterior of individual landmarks within each 2D bounding box $P(p_k|I, \hat{g}_b)$. We use the mean-square error as loss.

$$\frac{1}{K}\sum_{k=1}^{K}||\hat{w}_{k,i} - w_{k,i}||^2 \tag{2.12}$$

where $\hat{w}_{k,i} \in \mathbb{R}^{64\times64}$ is the predicted heat map for the k_{th} landmark and $w_{k,i} \in \mathbb{R}^{64\times64}$ is a 2D Gaussian with standard deviation of 1 pixel centered at the k_{th} landmark ground truth location. When a landmark is occluded or out of view, its ground truth heat map is set to all zeros. Each vehicle is modeled by 14 landmarks.

3D Orientation and scale hypotheses. $P(\theta, \sigma|I, \hat{g}_b)$ is approximated by a deep network

with ResNet backbone [HZR16], yielding $Q(\hat{g}_B|I, \hat{g}_b)$ where $I_{|_{\hat{g}_b}}$ is a (64×64) crop of the (centered and re-scaled) image $I \circ \hat{g}_b{}^{-1}$. We design a joint loss with multi-scale supervision for training pose and 3D scales.

$$H_{P,Q}(\theta_j, \hat{\theta}_i) + \lambda d(\sigma_j, \hat{\sigma}_i) \tag{2.13}$$

where Q is an approximation of $P(\theta|I, \hat{g}_b)$, both of which are Von Mises distributions over the discrete set $\{0, \ldots, 359^o\}$. We use a cross-entropy loss for orientation estimation. At inference time, the MAP estimate $\hat{\theta}_i \in [0, 2\pi)$ is used as a vehicle's azimuth estimation. $P(\sigma|I, \hat{g}_b, \theta)$ is estimated jointly with azimuth in the same network using the L^1 distance $d(\sigma_j, \hat{\sigma}_i)$. Empirically, we found that imposing the orientation loss on the intermediate feature map and the size loss on the last layer generates better results than minimizing both losses on the same layer.

Shape hypotheses. The 3D morphable shape model is learnt using 2D landmarks via an EM-Gaussian method [THB03, KTC15]. The hypothesis is that 3D object shapes are confined to a low-dimensional basis of the entire possible shape space. Therefore, the normalized 3D shape $S_m \in \mathbb{R}^{3K\times 1}$ of each vehicle instance can be factorized as the sum of the mean shape $\bar{S} \in \mathbb{R}^{3K\times 1}$ of this category deformed using a linear combination of N basis shapes, $V_n \in \mathbb{R}^{3K\times 1}$. For each car, the orthographic projection constraint between its 3D shape S_m and 2D landmarks $p_{k,m}$ is constructed as:

$$p_{k,m} = c_m R_m (P_{k,m} + t_m) + \zeta_{k,m} \tag{2.14}$$

$$S_m = \bar{S} + \sum_{n=1}^{N} \alpha_{n,m} V_n \tag{2.15}$$

$$\zeta_{k,m} \sim N(0, \sigma^2 I_{2\times 2}), \quad \alpha_{n,m} \sim N(0, 1), \quad R_m^T R_m = I_{3\times 3} \tag{2.16}$$

in which $c_m \in \mathbb{R}^{2\times 2}$ is the scaling factor of the orthography (para-perspective projection). $R_m \in \mathbb{R}^{2\times 3}$ and $t_m \in \mathbb{R}^{3\times 1}$ are the orthographic rotation and translation of each object instance in the camera coordinate, respectively. $P_{k,m} \in \mathbb{R}^{3\times 1}$ represents the k_{th} 3D landmark in S_m.

Monocular depth. $P(Z|I)$ is learned from stereo disparity, converted to depth using camera calibration. Similar to vehicle landmark detection, we use an hourglass network, with skip-connections, to predict per-pixel disparity [GAB17]. The left view is input to the encoder, and the right view used as supervision for appearance matching at multiple output scales of the decoder. The total loss is accumulated across four output scales combining three terms. One measures the quality of image matching, one measures disparity smoothness, and the last measures left/right disparity consistency. Next we describe each term relative to the left view. Each term is replicated for the right view to enforce left-view consistency. Appearance is measured by

$$\frac{1}{D}\sum_{a,b}\beta\frac{1-SSIM(I^l_{ab},\widetilde{I}^l_{ab})}{2}+(1-\beta)||I^l_{ab}-\widetilde{I}^l_{ab}||_1 \tag{2.17}$$

which combines single-scale SSIM [WBS04] and the L^1 distance between the input image I^l and its reconstruction $\widetilde{I}^l$ obtained by warping I^r using the disparity d^r with a differentiable image sampler from the spatial transformer network [JSZ15]. Disparity smoothness is measured by

$$\frac{1}{D}\sum_{a,b}|\partial_x d^l_{ab}|e^{-||\partial_x I^l_{ab}||}+|\partial_y d^l_{ab}|e^{-||\partial_y I^l_{ab}||} \tag{2.18}$$

which contains an edge-aware term depending ∂I [HKJ13]. Finally, left/right consistency is measured using the L^1 norm.

$$\frac{1}{D}\sum_{a,b}\|d^l_{ab}-d^r_{ab-d^l_{ab}}\|_1. \tag{2.19}$$

2.4 Implementation Details

It takes about one week to train the 2D bounding box network, and two hours for the orientation/3D scale network on KITTI with 4 TITAN-X GPUs. The landmark detector is trained on Pascal3D. The training process for the monocular depth estimation network is unsupervised using KITTI stereo-pairs, which takes around 5 to 12 hours depending on the amount of data available. In theory, these deep networks could be unified into a single

Method	3DVP	3DOP	Mono3D	GoogLenet DeepMANTA	VGG16 DeepMANTA	Mono3D++
Type	Mono	Stereo	Mono	Mono	Mono	Mono
Time	40 s	3 s	4.2 s	0.7 s	2 s	**0.6 s**

Table 2.1: Inference time comparisons against a paragon set of both monocular and stereo methods.

	1 meter			2 meters			3 meters		
Method	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
3DVP	45.6 / -	34.3 / -	27.7 / -	65.7 / -	54.6 / -	45.6 / -	- / -	- / -	- / -
SubCNN	39.3 / -	31.0 / -	26.0 / -	70.5 / -	56.2 / -	47.0 / -	- / -	- / -	- / -
Mono3D	- / 46.0	- / 38.3	- / 34.0	- / 71.0	- / 59.9	- / 53.8	- / 80.3	- / 69.3	- / 62.7
GoogLenet DeepMANTA	70.9 / 65.7	58.1 / 53.8	49.0 / 47.2	90.1 / 89.3	77.0 / 75.9	66.1 / 67.3	- / -	- / -	- / -
VGG16 DeepMANTA	66.9 / 69.7	53.2 / 54.4	44.4 / 47.8	88.3 / 91.0	74.3 / 76.4	63.6 / 67.8	- / -	- / -	- / -
Mono3D++	**80.6** / **80.2**	**67.7** / **65.1**	**56.0** / **54.6**	**93.3** / **92.7**	**83.0** / **80.8**	**71.8** / **70.5**	**95.0** / **95.4**	**86.7** / **85.4**	**76.2** / **75.9**

Table 2.2: 3D localization comparisons with monocular methods on KITTI val1/val2 by ALP of 1, 2 and 3 meters thresholds for 3D box center distance.

		1 / 2 / 3 meters		
Method	Type	Easy	Moderate	Hard
Mono3D++	Mono	80.2 / 92.7 / 95.4	65.1 / 80.8 / 85.4	54.6 / 70.5 / 75.9
3DOP	Stereo	78.6 / 87.4 / 89.5	66.9 / 80.0 / 84.2	59.4 / 71.9 / 76.0

Table 2.3: 3D localization comparisons with the state-of-the-art stereo method by ALP under 3D box center distance thresholds of 1, 2 and 3 meters. Note that our method only uses a single image for inference, while 3DOP needs stereo-pairs.

one and trained jointly, but this is beyond our scope here. Learning the morphable shape model takes about 2.5 minutes using 2D vehicle landmarks. At inference time, we use the Ceres solver [AMO] to optimize the weighted loss Eq. (2.10). On average it converges in 1.5 milliseconds within about 15 iterations. Detailed timing comparisons are shown in Tab. 2.1.

2.5 Experiments

We evaluate our method on the KITTI object detection benchmark. This dataset contains $7,481$ training images and $7,518$ test images. To facilitate comparison with competing approaches, we isolate a validation set from the training set according to the same protocol of [XCL15, XCL17, CCR17] called (train1, val1), and the same used by [CKZ15, CKZ16, CCR17] called (train2, val2). We report results using five evaluation metrics: three for 3D and two for 2D localization. For the former, we use average localization precision (ALP) [XCL15], average precision based on 3D intersection-over-union (IoU), AP_{3D}, and bird's eye view based average localization precision, AP_{loc} [GLU12]. Although 2D localization is not our goal, as a sanity check we also measure average precision (AP) and average orientation similarity (AOS).

ALP is based on the distance between the center of the detected 3D boxes and the annotated ground truth. A 3D detection is correct if the distance is below a threshold. AP_{3D} is computed from the IoU of 3D bounding boxes. AP_{loc} is obtained by projecting the 3D bounding boxes to the ground plane (bird's eye view) and computing the 2D IoU with ground truth. Note that while ALP only measures distance between centers, both AP_{3D} and AP_{loc} jointly evaluate a vehicle's translation, orientation and 3D scales. AOS measures 2D orientation relative to ground truth.

The paragon set for our experiments consists of 3DVP [XCL15], SubCNN [XCL17], Mono3D [CKZ16], 3DOP [CKZ15] and DeepMANTA [CCR17]. While 3DOP is the state-of-the-art stereo method, the rest are monocular. Among the monocular methods, DeepMANTA is the current state-of-the-art. Also related to our method for monocular 3D vehicle detection is Deep3DBox [MAF17], which however used different evaluation metrics from the ones above, thus preventing direct comparison. For object 2D orientation and bounding box evaluation, we also compare with Faster-RCNN [RHG15] as well as Deep3DBox.

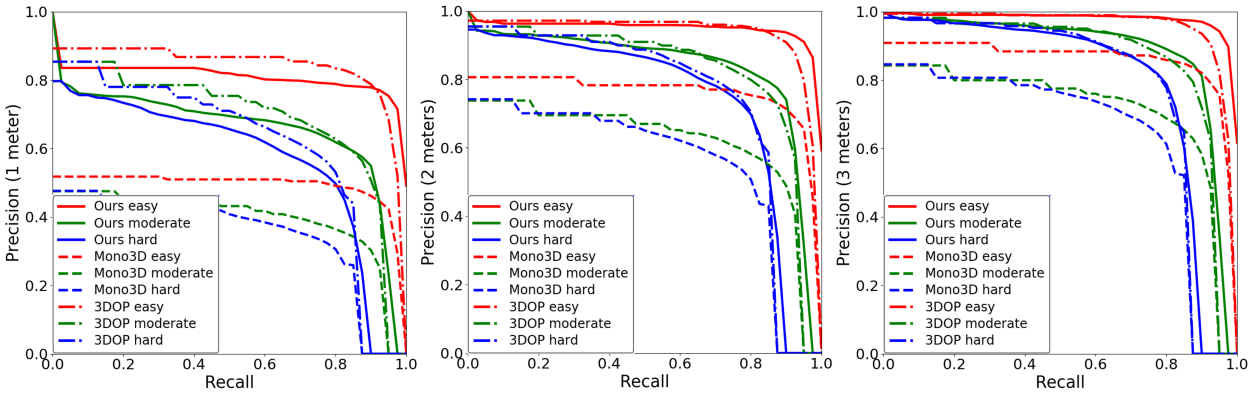


Figure 2.4: Recall/3D localization precision curves for 1 meter (left), 2 meters (middle) and 3 meters (right) precision on KITTI val2. Solid lines are our results. Dashed lines are Mono3D. Dash-dot lines are 3DOP, which uses stereo-pairs at inference time.

2.5.1 3D localization

We use ALP with distance thresholds of 1, 2 and 3 meters in Tab. 2.2 including both val1 and val2. Our method improves the state-of-the-art monocular method, DeepMANTA, by 10.5% on average. Even though our method is monocular, we compare to the stereo method 3DOP using val2 in Tab. 2.3. Surprisingly, we outperform 3DOP on “easy" and “moderate” cases and is comparable on “hard" case. Detailed comparisons by precision/recall curves are shown in Fig. 2.4. We outperform the monocular by large margins and is better than the stereo in some cases.

2.5.2 3D detection and birds’ eye view localization

We use AP_{3D} with 3D IoU thresholds of 0.25, 0.5 and 0.7, as well as AP_{loc} with 2D IoU thresholds of 0.5 and 0.7. Tab. 2.4 shows comparisons with both monocular and stereo methods using val2. Our method surpasses all monocular ones uniformly. Though the monocular setting is more challenging than the stereo one due to the lack of depth, our method still outperforms 3DOP by about 39% to 61% on AP_{3D} with IoU threshold of 0.7. Tab. 2.5 shows comparison on AP_{loc} with both monocular and stereo using val2. Again, our

		3D IoU 0.25 / 0.50 / 0.70		
Method	Type	Easy	Moderate	Hard
Mono3D	Mono	62.9 / 25.2 / 2.5	48.2 / 18.2 / 2.3	42.7 / 15.5 / 2.3
Mono3D++	Mono	**71.9** / **42.0** / **10.6**	**59.1** / **29.8** / **7.9**	**50.5** / **24.2** / **5.7**
3DOP	Stereo	85.5 / 46.0 / 6.6	68.8 / 34.6 / 5.1	64.1 / 30.1 / 4.1

Table 2.4: Comparisons on AP_{3D} under different 3D IoU thresholds with both monocular and stereo methods.

		2D IoU 0.50 / 0.70		
Method	Type	Easy	Moderate	Hard
Mono3D	Mono	30.5 / 5.2	22.4 / 5.2	19.2 / 4.1
Mono3D++	Mono	**46.7** / **16.7**	**34.3** / **11.5**	**28.1** / **10.1**
3DOP	Stereo	55.0 / 12.6	41.3 / 9.5	34.6 / 7.6

Table 2.5: Comparisons on AP_{loc} under different 2D IoU thresholds with both monocular and stereo methods.

results surpass monocular ones uniformly. Even if compared with the stereo method, we gain around 21% to 33% relative improvement on AP_{loc} under 0.7 IoU.

2.5.3 Ablation studies

In Tab. 2.6, 2.7 and 2.8 we use val1 and val2 with ALP, AP_{3D} and AP_{loc} to validate our joint modeling of the coarse and the fine-scaled 3D hypotheses, as well as task priors. "v1" indicates our inference scheme at initialization; "v2" only models the coarse geometric innovation and a ground plane constraint; "v3" adds the fine-scaled geometric innovation and vehicle shape priors. Best results are achieved by our overall model "v4", which further considers unsupervised monocular depth. Due to the page limit, extended comparisons over different threshold values are reported in Sec. 2.7.

Method	Easy	Moderate	Hard
v1	13.6 / 16.9	12.2 / 13.3	11.3 / 12.5
v2	68.5 / 68.2	58.3 / 57.5	50.8 / 47.6
v3	76.1 / 73.2	64.5 / 60.2	53.6 / 50.0
v4	**80.6** / **80.2**	**67.7** / **65.1**	**56.0** / **54.6**

Table 2.6: Ablation studies on val1/val2 by ALP under 3D box center distance threshold of 1 meter.

Method	Easy	Moderate	Hard
v1	10.50 / 11.50	8.75 / 8.99	9.02 / 16.43
v2	68.33 / 58.50	55.00 / 49.96	49.17 / 45.09
v3	71.39 / 66.59	59.06 / 54.88	50.59 / 48.26
v4	**79.45** / **71.86**	**62.76** / **59.11**	**52.79** / **50.53**

Table 2.7: Ablation studies on val1/val2 by AP_{3D} with 3D IoU threshold of 0.25.

Method	Easy	Moderate	Hard
v1	2.06 / 2.27	2.30 / 2.27	2.29 / 2.36
v2	37.27 / 30.18	27.48 / 24.82	23.67 / 21.49
v3	42.68 / 37.25	32.12 / 28.50	25.84 / 24.14
v4	**50.50** / **46.68**	**36.85** / **34.32**	**29.05** / **28.13**

Table 2.8: Ablation studies on val1/val2 by AP_{loc} under 2D IoU threshold of 0.5.

2.5.4 2D detection and orientation

As a sanity check, the 2D detection AP and AOS are also evaluated with monocular and stereo methods. Our estimation is on par with the state-of-the-art results. Detailed comparisons are included in Sec. 2.7.

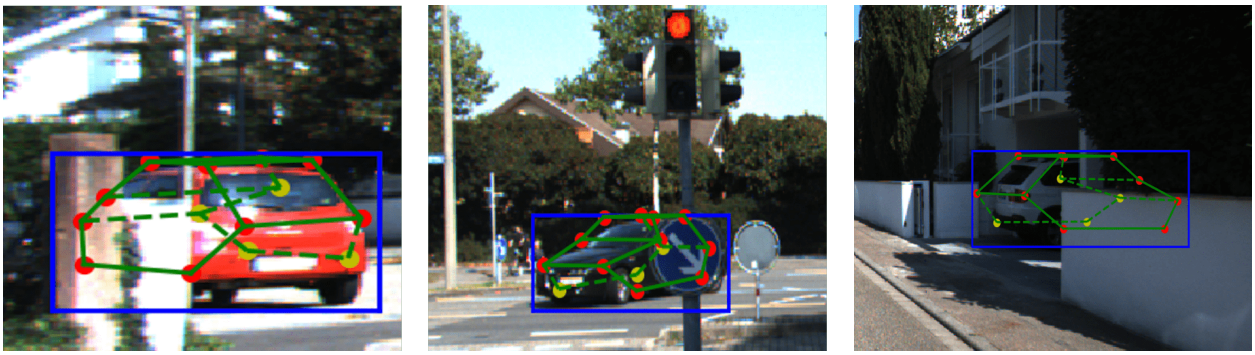


Figure 2.5: Typical issues (e.g. partial occlusions, shadow, low resolution) that affect 2D vehicle landmark measurements.

2.5.5 Qualitative results

Fig. 2.1 shows representative outputs of our method, including cars at different scales, 3D shapes, poses and occlusion patterns. Fig. 2.5 illustrates typical issues addressed by jointly modeling a vehicle's coarse and fine-scaled 3D shape and pose representations. When vehicle landmarks suffer from partial occlusions, shadow or low resolution, we can still leverage 2D bounding boxes in order to enforce the two-scale geometric innovation constraints.

2.5.6 Generalization

Although not our focus here, chairs share similar constraints to vehicles like the two-scale 3D hypotheses innovation, a ground plane assumption, shape priors, etc. Thus to demonstrate the generality of our method, we also test on chairs and report results in Sec. 2.7.

2.5.7 Failure modes

In Fig. 2.6 we illustrate some failure cases, which include field of view truncation, causing the bounding box projection constraint E_{2D3D} in the overall energy to enforce the 3D bounding box's 2D projection to be within the truncated 2D box measurement. Failures can also occur due to inaccurate orientation estimation, and under-representation in the training set

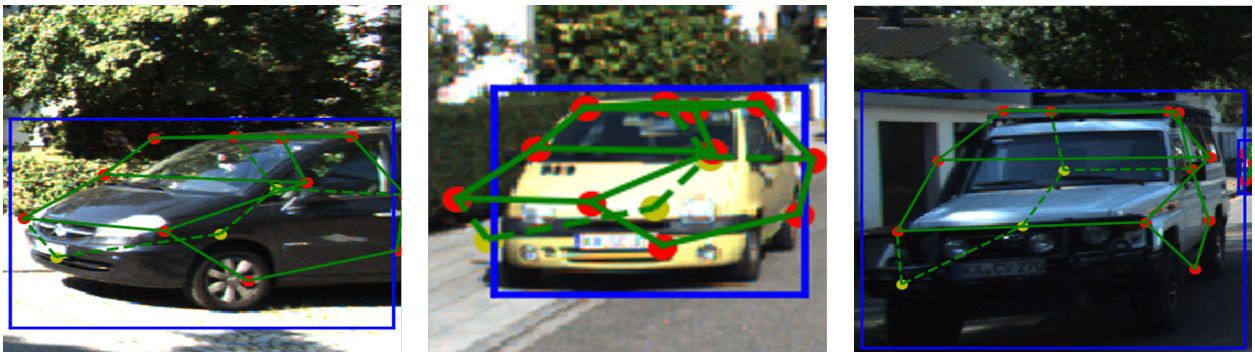


Figure 2.6: Failure cases caused by field of view truncation, inaccurate orientation or 3D scale estimation.

(oversized SUV) which causes the normalized morphable wireframe model to be rescaled by incorrect 3D size estimation.

2.6 Discussion

We have presented a method to infer vehicle pose and shape in 3D from a single RGB image that considers both the coarse and the fine-scaled 3D hypotheses, and multiple task priors, as reflected in an overall energy function Eq. (2.10) for joint optimization. Our inference scheme leverages independence assumptions to decompose the posterior probability of pose and shape given an image into a number of factors. For each term we design a loss function that is initialized by output from deep network as shown in Fig. 2.3. Our method improves the state-of-the-art for monocular 3D vehicle detection under various evaluation settings.

2.7 Additional Results

2.7.1 More qualitative demos

In Fig. 2.9, Fig. 2.10 and Fig. 2.11 we show 3D vehicle detection results on KITTI test set, val1 and val2, respectively. Moreover, results on chairs are shown in Fig. 2.8 in order to

demonstrate the generality of our method.

2.7.2 More quantitative benchmarks

Ablation Studies. In Tab. 2.10, 2.11 and 2.12 we use val1 and val2 with ALP, AP_{3D} and AP_{loc} to validate our joint modeling of the coarse and the fine-scaled 3D hypotheses, as well as task priors. "v1" indicates our inference scheme at initialization; "v2" only models the coarse geometric innovation and a ground plane constraint; "v3" adds the fine-scaled geometric innovation and vehicle shape priors. Best results are achieved by our overall model "v4", which further considers unsupervised monocular depth.

2D Detection and Orientation. As a sanity check, in Tab. 2.13 the 2D detection AP and AOS are evaluated on KITTI test set with monocular and stereo methods. Our results are on par with the state-of-the-art results.

2.7.3 Morphable wireframe model

Each vehicle instance is represented by 14 landmarks. In Tab. 2.9, we explain the landmarks. In Fig. 2.7, we visualize the normalized morphable wireframe model under three types of deformations.

2.7.4 Modeling assumptions

2.7.4.1 Overview

The goal is to approximate $P(g_B, S|I)$. Maximizing this posterior is equivalent to minimizing the negative log as below:

$$-log[P(g_b|g_B)P(p|g_B,S)P(Z_b|T_Z)P(T)P(S)] \tag{2.20}$$

For notation definitions, please refer to the main paper. The corresponding overall energy function is defined as:

$$E = E_{2D3D}(g_b, g_B) + \lambda_1 E_{LP}(p, g_B, P) + \lambda_2 E_{MD}(Z_b, T) + \lambda_3 E_{GP}(T) + \lambda_4 E_S(\alpha) \tag{2.21}$$

Vehicle Part	Notation	Landmark Position
Wheels	w_{lf}	left-front wheel center
	w_{rf}	right-front wheel center
	w_{lt}	left-tail wheel center
	w_{rt}	right-tail wheel center
Headlights	h_l	left headlight center
	h_r	right headlight center
Taillights	t_l	left taillight center
	t_r	right taillight center
Side mirrors	m_l	left-side mirror
	m_r	right-side mirror
Roof corners	r_{lf}	left-front roof corner
	r_{rf}	right-front roof corner
	r_{lt}	left-tail roof corner
	r_{rt}	right-tail roof corner

Table 2.9: The 14 landmarks that constitute the 3D morphable wireframe model.

where the first two terms indicate forward projection errors of the coarse and the fine-scaled 3D hypotheses. The last three terms, respectively, represent constraints enforced via unsupervised monocular depth, a ground plane assumption, and vehicle shape priors.

2.7.4.2 Derivation

We start with the goal, which is the posterior $P(g_B, S|I)$, and marginalize cars' 2D bounding box g_b.

$$\int P(g_B, S|I, g_b)P(g_b|I)d_{g_b} \tag{2.22}$$

where we use a 2D detection deep network to approximate the second term in the integration. Note that g_B consists of (θ, σ, T). Thus the first term can be written as:

$$P(\theta, \sigma|I, g_b)P(T, S|I, g_b, \theta, \sigma) \tag{2.23}$$

in which the first term is approximated by a deep network that jointly estimates vehicle orientation and 3D scale. Next, we marginalize the second term with respect to the 2D

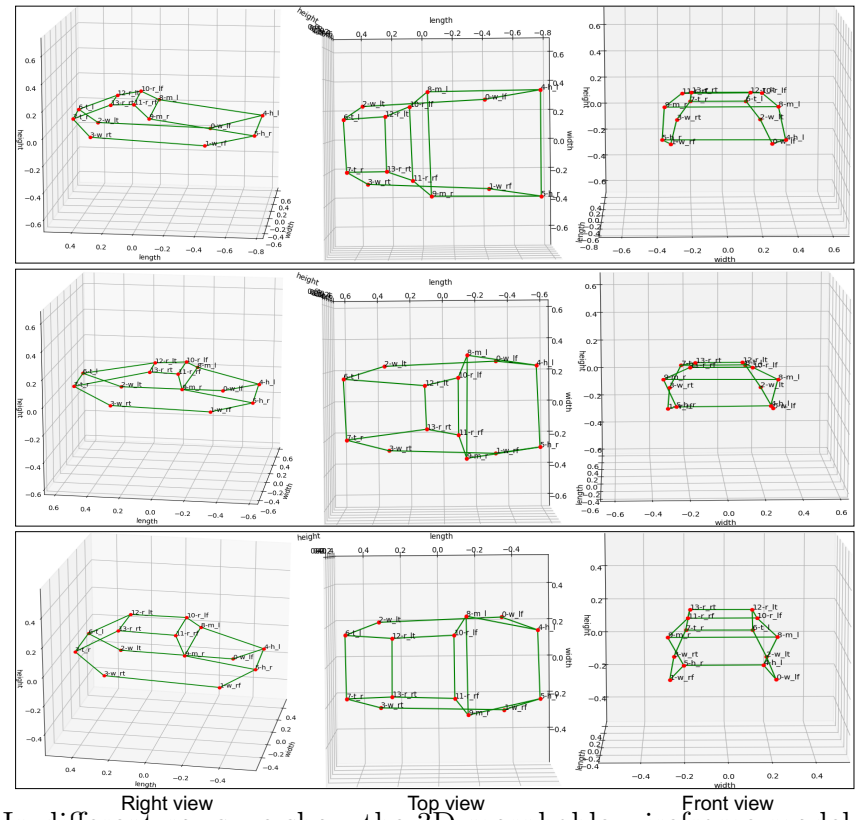


Figure 2.7: In different rows we show the 3D morphable wireframe model under different types of deformations. The left column is the right view. The middle column is the top view. The right column is the front view.

landmarks, assuming that given a cropped image, the 2D landmarks are independent of the azimuth and the scale.

$$\int P(T, S|I, g_b, \theta, \sigma, p)P(p|I, g_b)d_p \tag{2.24}$$

The second term in the integration is approximated by a landmark detection network. The first term will be marginalized with respect to the average depth Z_b of a cropped image specified by I and g_b. We assume that given a cropped image, Z_b is independent of (θ, σ, p).

$$\int P(T, S|I, g_b, \theta, \sigma, p, Z_b)P(Z_b|I, g_b)d_{Z_b} \tag{2.25}$$

where we use an unsupervised monocular depth estimation network to approximate the second term in the integration. Now we apply Bayes' rule to the first term.

$$P(T, S|I, g_b, \theta, \sigma, p, Z_b) = \frac{P(T, S, I, g_b, \theta, \sigma, p, Z_b)}{P(I, g_b, \theta, \sigma, p, Z_b)} \tag{2.26}$$

Since the goal here is to maximize the posterior, multiplicative factors (e.g. the denominator) that do not depend on (T, S) do not affect the solution and can be ignored in the optimization. To decompose the numerator into a tractable form, we make following assumptions:

- The image I, the azimuth θ and the scale σ all follow uniform prior distribution. The numerator of Eq. (2.26) can then be written as:

$$P(I)P(\theta)P(\sigma)P(T, S, g_b, p, Z_b|I, \theta, \sigma) \tag{2.27}$$

- The translation T has ground plane prior.

$$P(I)P(\theta)P(\sigma)P(T)P(S, g_b, p, Z_b|I, \theta, \sigma, T) \tag{2.28}$$

- The shape S has minimum deformation prior.

$$P(I)P(\theta)P(\sigma)P(T)P(S)P(g_b, p, Z_b|I, \theta, \sigma, T, S) \tag{2.29}$$

- Given the 3D bounding box g_B, the 2D bounding box g_b is independent of the image I and the shape S.

$$P(I)P(\theta)P(\sigma)P(T)P(S)P(g_b|g_B)P(p, Z_b|I, \theta, \sigma, T, S, g_b) \tag{2.30}$$

- Given the 3D bounding box g_B and the shape S, the 2D landmarks p are independent of the image I and the 2D bounding box g_b.

$$P(I)P(\theta)P(\sigma)P(T)P(S)P(g_b|g_B)P(p|g_B, S)P(Z_b|I, \theta, \sigma, T, S, g_b, p) \tag{2.31}$$

- Given the position of the bounding box along the Z-axis, T_Z, the average depth Z_b of that vehicle's reprojection region is independent of the rest of the variables.

$$P(I)P(\theta)P(\sigma)P(T)P(S)P(g_b|g_B)P(p|g_B, S)P(Z_b|T_Z) \tag{2.32}$$

Based on the assumptions above, maximizing Eq. (2.32) is equivalent to minimizing the negative log in Eq. (2.20).

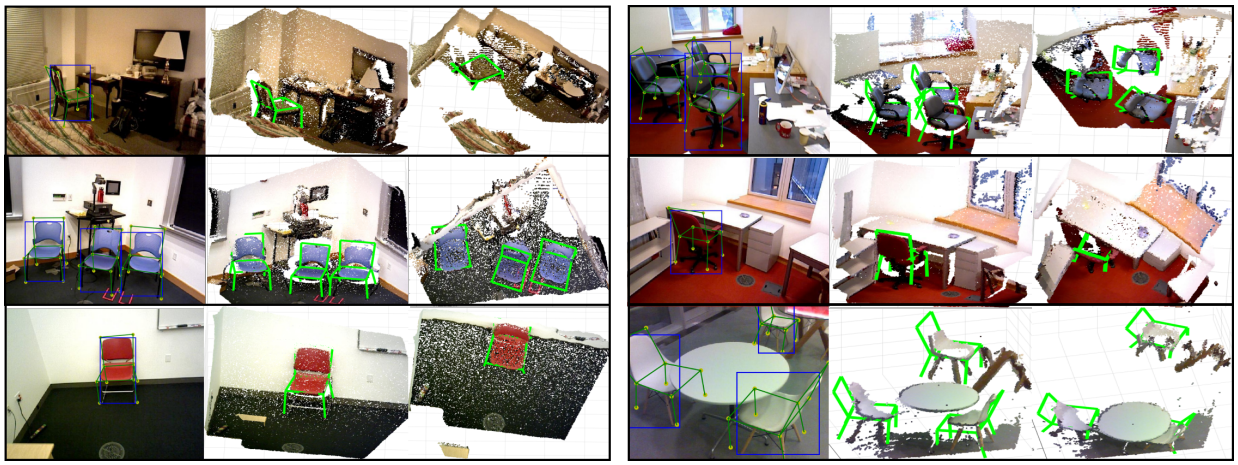


Figure 2.8: 3D detection on chairs in the SUN-RGBD dataset. Depth is only used for visualization. 3D chair skeletons in green. 2D bounding boxes in blue.

2.7.4.3 Optimization

We have decomposed the posterior of pose and shape given an image into a number of factors:

$$\iiint P(g_b|I)P(\theta,\sigma|I,g_b)P(p|I,g_b)P(Z_b|I,g_b)P(T,S|I,g_b,\theta,\sigma,p,Z_b)d_{g_b}d_p d_{Z_b} \tag{2.33}$$

in which a 2D detection network is used to approximate $P(g_b|I)$. $P(\theta,\sigma|I,g_b)$ is estimated via a deep network which jointly predicts vehicle orientation and 3D scale and $P(p|I,g_b)$ is learned by a landmark detection network. For $P(Z_b|I,g_b)$, we use an unsupervised monocular depth estimation network. The network outcome is used as initialization for maximizing the last posterior in the integration Eq. (2.33), which is equivalent to minimizing an overall energy function in Eq. (2.21). We apply a Cholesky factorization based solver to solve this nonlinear optimization problem. In theory, the same energy could be optimized end-to-end to fine-tune the weights of every neural network component, but this is beyond our scope here.

Figure 2.9: 3D detection results on KITTI test set, shown as projections on the input images. 3D morphable shape models of each vehicle are colored in green. Dashed green lines are occluded edges. For the 14 vertices of each morphable shape model, visible vertices are colored in red and occluded ones in yellow. 2D bounding boxes are colored in blue.

Figure 2.10: 3D detection results on KITTI val1.

Figure 2.11: 3D detection results on KITTI val2.

	1 meter			2 meters			3 meters		
Method	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
v1	13.6 / 16.9	12.2 / 13.3	11.3 / 12.5	40.6 / 43.6	33.7 / 34.0	30.5 / 36.3	63.3 / 67.7	52.7 / 53.3	51.0 / 49.3
v2	68.5 / 68.2	58.3 / 57.5	50.8 / 47.6	89.4 / 87.8	79.8 / 76.0	69.5 / 66.5	93.8 / 93.9	85.2 / 83.6	75.5 / 74.4
v3	76.1 / 73.2	64.5 / 60.2	53.6 / 50.0	91.2 / 90.5	81.1 / 78.4	70.5 / 68.7	94.0 / 94.4	85.3 / 84.7	75.7 / 75.2
v4	**80.6** / **80.2**	**67.7** / **65.1**	**56.0** / **54.6**	**93.3** / **92.7**	**83.0** / **80.8**	**71.8** / **70.5**	**95.0** / **95.4**	**86.7** / **85.4**	**76.2** / **75.9**

Table 2.10: Ablation studies on val1/val2 by ALP with multiple 3D box center distance thresholds.

	3D IoU=0.25			3D IoU=0.5			3D IoU=0.7		
Method	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
v1	10.50 / 11.50	8.75 / 8.99	9.02 / 16.43	1.27 / 1.59	1.05 / 1.55	0.98 / 1.29	0.76 / 0.38	0.70 / 0.38	0.70 / 0.38
v2	68.33 / 58.50	55.00 / 49.96	49.17 / 45.09	31.13 / 25.86	24.49 / 21.21	19.89 / 17.66	7.30 / 8.11	5.44 / 6.51	5.11 / 6.40
v3	71.39 / 66.59	59.06 / 54.88	50.59 / 48.26	38.55 / 33.66	27.92 / 24.79	22.12 / 19.97	10.24 / 7.72	7.27 / 5.45	6.10 / 4.83
v4	**79.45** / **71.86**	**62.76** / **59.11**	**52.79** / **50.53**	**43.09** / **41.98**	**30.53** / **29.77**	**23.96** / **24.23**	**14.42** / **10.64**	**10.75** / **7.89**	**8.03** / **5.66**

Table 2.11: Ablation studies on val1/val2 by AP_{3D} with multiple 3D IoU thresholds.

	2D IoU=0.5			2D IoU=0.7		
Method	Easy	Moderate	Hard	Easy	Moderate	Hard
v1	2.06 / 2.27	2.30 / 2.27	2.29 / 2.36	0.76 / 0.40	0.70 / 0.42	0.70 / 0.41
v2	37.27 / 30.18	27.48 / 24.82	23.67 / 21.49	11.40 / 11.40	9.05 / 9.52	7.29 / 8.03
v3	42.68 / 37.25	32.12 / 28.50	25.84 / 24.14	14.25 / 10.92	10.79 / 8.86	8.94 / 6.94
v4	**50.50** / **46.68**	**36.85** / **34.32**	**29.05** / **28.13**	**20.21** / **16.71**	**13.63** / **11.47**	**11.24** / **10.12**

Table 2.12: Ablation studies on val1/val2 by AP_{loc} with multiple 2D IoU thresholds.

		AP			AOS		
Method	Type	Easy	Moderate	Hard	Easy	Moderate	Hard
3DVP	Mono	81.46	75.77	65.38	81.02	74.59	64.11
Faster-RCNN	Mono	87.90	79.11	70.19	-	-	-
SubCNN	Mono	**90.75**	**88.86**	**79.24**	**90.61**	88.43	**78.63**
3DOP	Stereo	90.09	88.34	78.79	88.56	85.81	76.21
Mono3D	Mono	90.27	87.86	78.09	89.00	85.83	76.00
Deep3DBox	Mono	90.47	88.86	77.60	90.39	**88.56**	77.17
DeepMANTA	Mono	**97.25**	**90.03**	**80.62**	**97.19**	**89.86**	**80.39**
Mono3D++	Mono	**90.59**	**89.88**	**80.61**	**90.41**	**89.23**	**79.60**

Table 2.13: Comparisons with monocular and stereo methods by 2D detection AP and AOS on KITTI test set.

CHAPTER 3

Point Cloud Deep Geodesic Representation Learning

3.1 Introduction

Determining neighborhood relationship among points in a point cloud, known as topology estimation, is an important problem since it indicates the underlying point cloud structure, which could further reveal the point cloud semantics and functionality. Consider the red inset in Fig. 3.1: the two clusters of points, though seemingly disconnected, should indeed be connected to form a chair leg, which supports the whole chair. On the other hand, the points on opposite sides of a chair seat, though spatially very close to each other, should not be connected to avoid confusing the sittable upper surface with the unsittable lower side. Determining such topology appears to be a very low-level endeavor but in reality it requires global, high-level knowledge, making it a very challenging task. Still, from the red inset in Fig. 3.1, we could draw the conclusion that the two stumps are connected only after we learn statistical regularities from a large number of point clouds and observe many objects of this type with connected elongated vertical elements extending from the body to the ground. This motivates us to adopt a learning approach to capture the topological structure within point clouds.

Our primary goals in this paper are to develop representations of point cloud data that are informed by the underlying surface topology as well as object geometry, and propose methods that leverage the learned topological features for geodesic-aware point cloud analysis. The representation should capture various topological patterns of a point cloud and the method

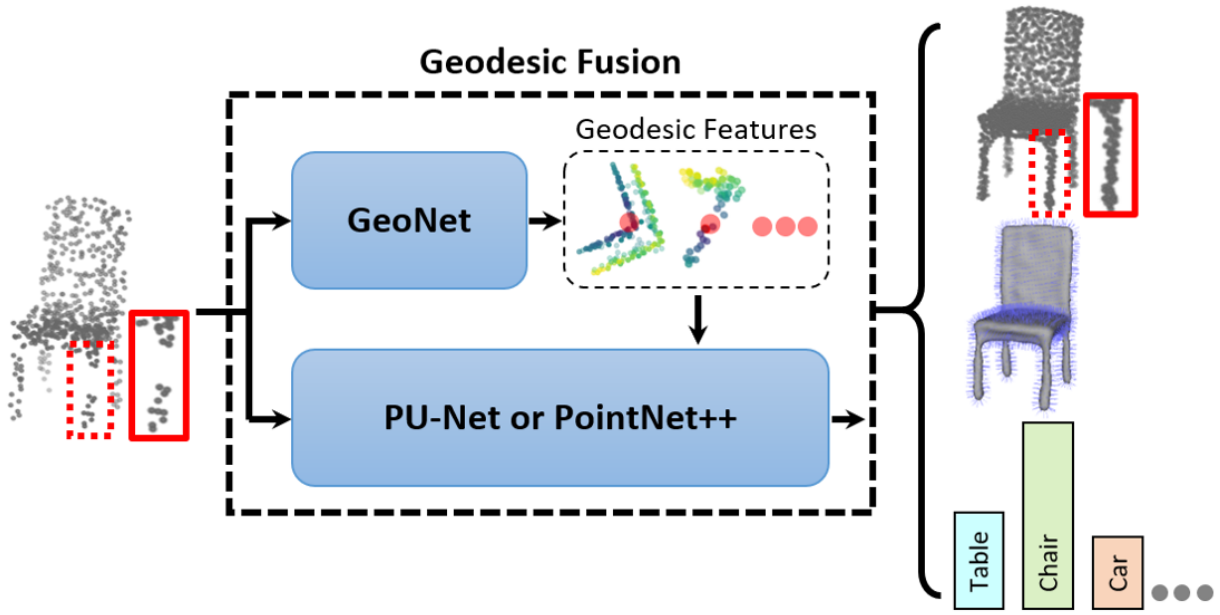


Figure 3.1: Our method takes a point cloud as input, and outputs representations used for multiple tasks including upsampling, normal estimation, mesh reconstruction, and shape classification.

of leveraging these geodesic features should not alter the data stream, so our representation can be learned jointly and used in conjunction with the state-of-the-art baseline or backbone models (e.g. PU-Net, PointNet++ [YLF18, QSM17a, QYS17, CHL18]) that feed the raw data through, with no information loss to further stages of processing.

For the first goal, we propose a geodesic neighborhood estimation network (GeoNet) to learn deep geodesic representations using the ground truth geodesic distance as supervision signals. As illustrated in Fig. 3.2, GeoNet consists of two modules: an autoencoder that extracts a feature vector for each point and a geodesic matching (GM) layer that acts as a learned kernel function for estimating geodesic neighborhoods using the latent features. Due to the supervised geodesic training process, intermediate features of the GM layer contain rich information of the point cloud topology and intrinsic surface attributes. We note that the representation, while trained on geodesic distances, does not by construction produce

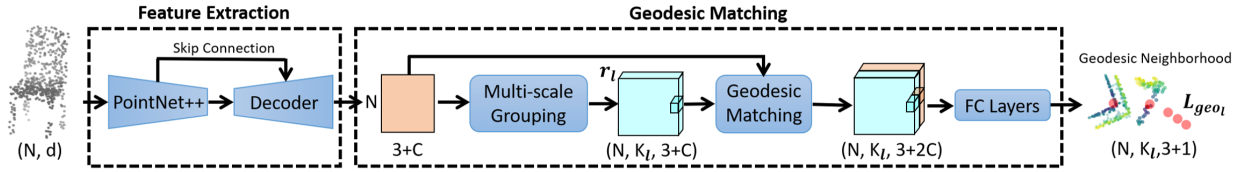


Figure 3.2: GeoNet: geodesic neighborhood estimation network.

geodesics (e.g. symmetry, triangle inequality, etc.). The goal of the representation is to inform subsequent stages of processing of the global geometry and topology, and is not to conduct metric computations directly.

For the second task, as shown in Fig. 3.3, we propose geodesic fusion schemes to integrate GeoNet into the state-of-the-art network architectures designed for different tasks. Specifically, we present PU-Net fusion (PUF) for point cloud upsampling, and PointNet++ fusion (POF) for normal estimation, mesh reconstruction as well as non-rigid shape classification. Through experiments, we demonstrate that the learned geodesic representations from GeoNet are beneficial for both geometric and semantic point cloud analyses.

In summary, in this paper we propose an approach for learning deep geodesic-aware representations from point clouds and leverage the results for various point set analyses. Our contributions are:

- We present, to the best of our knowledge, the first deep learning method, GeoNet, that ingests point clouds and learns representations which are informed by the intrinsic structure of the underlying point set surfaces.
- To demonstrate the applicability of learned geodesic representations, we develop network fusion architectures that incorporate GeoNet with baseline or backbone networks for geodesic-aware point set analysis.
- Our geodesic fusion methods are benchmarked on multiple geometric and semantic point set tasks using standard datasets and outperform the state-of-the-art methods.

3.2 Related Work

We assume our input is a point cloud, which can be obtained by multiple-view geometry, single image depth estimation [MSK12, HZ03, GAB17, WHS19, FWS19] or collected from various sensors (e.g. depth camera, Lidar, etc.). We mainly review traditional graph-based methods for geodesic distance computation, as well as general works on point cloud upsampling, normal estimation, and non-rigid shape classification, as we are unaware of other prior works on point cloud-based deep geodesic representation learning.

Geodesic distance computation. There are two types of methods: some allow the path to traverse mesh faces [SS86, MMP87, CH90, GH97, SSK05, XW09, CWW13] for accurate geodesic distance computation, while others find approximate solutions via shortest path algorithms constrained on graph edges [Dij59, Flo62, Joh77]. For the first type, an early method [SS86] suggests a polynomial algorithm of time $O(n^3 logn)$ where n is the number of edges, but their method is restricted to a convex polytope. Based on Dijkstra's algorithm [Dij59], [MMP87] improves the time complexity to $O(n^2 logn)$ and extends the method to an arbitrary polyhedral surface. Later, [CH90] proposes an $O(n^2)$ approach using a set of windows on the polyhedron edges to encode the structure of the shortest path set. By filtering out useless windows, [XW09] further speeds up the algorithm. Then [CWW13] introduces a heat method via solving a pair of standard linear elliptic problems. As for graph edge-based methods, typical solutions include Dijkstra's [Dij59], Floyd-Warshall [Flo62] and Johnson's algorithms [Joh77], which have much lower time complexity than the surface traversing methods. For a 20000-vertex mesh, computing its all-pair geodesic distances can take several days using [XW09] while [Joh77] only uses about 1 minute on CPU. When a mesh is dense, the edge-constrained shortest path methods generate low-error geodesic estimates. Thus in our work, we apply [Joh77] to compute the ground truth geodesic distance.

Point upsampling. Previous methods can be summarized into two categories. i) Optimization based methods [ABC03, LCL07, HLZ09], championed by [ABC03], which

interpolates a dense point set from vertices of a Voronoi diagram in the local tangent space. Then [LCL07] proposes a locally optimal projection (LOP) operator for point cloud resampling and mesh reconstruction leveraging an L_1 median. For improving robustness to point cloud density variations, [HLZ09] presents a weighted LOP. These methods all make strong assumptions, such as surface smoothness, and are not data-driven, and therefore have limited applications in practice. ii) Deep learning based methods. To apply the (graph) convolution operation, many of those methods first voxelize a point cloud into regular volumetric grids [WSK15, WZX16, HLH17, DQN17] or instead use a mesh [DBV16, YSG17]. While voxelization introduces discretization artifacts and generates low resolution voxels for computational efficiency, mesh data can not be trivially reconstructed from a sparse and noisy point cloud. In [YWS19, YS18] a sparse point cloud is reprojected onto a range map and modeled as a 2.5D inpanting problem. To directly upsample a 3D point cloud, PU-Net [YLF18] learns multilevel features for each point and expands the point set by a multibranch convolution unit in feature space. But PU-Net is based on Euclidean space and thus does not leverage the underlying point cloud surface attributes in geodesic space, which we show in this paper are important for upsampling.

Normal estimation. A widely used method for point cloud normal estimation is to analyze the variance in a tangential plane of a point and find the minimal variance direction by Principal Component Analysis (PCA) [HDD92, Jol11]. But this method is sensitive to the choice of the neighborhood size, namely, large regions can cause over-smoothed surfaces and small ones are sensitive to noises. To improve robustness, [GG07, CP05, AB99] propose to fit higher-order shapes. However, methods described above all require careful parameter tuning at the inference time and only estimate normal orientation up to sign. Thus, so far robust estimation for oriented normal vectors using traditional methods is still challenging, especially across different noise levels and shape structures. There are only few data-driven methods that are able to integrate normal estimation and orientation alignment into a unified pipeline [GKO18, QYS17]. They take a point cloud as input and directly regress

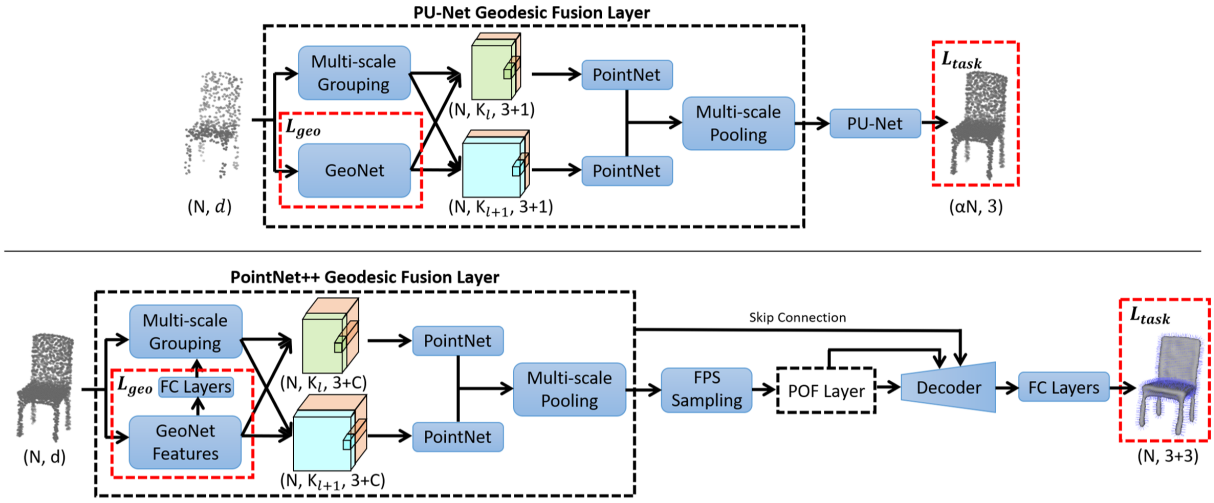


Figure 3.3: PU-Net (top) and PointNet++ (bottom) geodesic fusion architectures.

oriented normal vectors, but these methods are not designed to learn geodesic topology-based representations that capture the intrinsic surface features for better normal estimation.

Non-rigid shape classification. Classifying the point cloud of non-rigid objects often consists of two steps: extracting intrinsic features in geodesic space and applying a classifier (e.g. SVM, MLP, etc.). Some commonly used features include wave kernel signatures [ASC11], heat kernel signatures [SOG09], spectral graph wavelet signatures [MLH16], Shape-DNA [RWP06], etc. For example, DeepGM [LH18] uses geodesic moments and stacked sparse autoencoders to classify non-rigid shapes, such as cat, horse, spider, etc. The geodesic moments are feature vectors derived from the integral of the geodesic distance on a shape, while stacked sparse autoencoders are deep neural networks consisting of multiple layers of sparse autoencoders. However, the above methods all require knowing graph-based data, which is not available from widely used sensors (e.g. depth camera, Lidar, etc.) for 3D data acquisition. Though PointNet++ [QYS17] is able to directly ingest a point cloud and conduct classification, it is not designed to model the geodesic topology of non-rigid shapes and thus its performance is inferior to traditional two-step methods which heavily reply on the offline computed intrinsic surface features.

3.3 Methodology

3.3.1 Notation

$\chi = \{x_i\}$ denotes a point set with $x_i \in \mathbb{R}^d$ and $i = 1, \ldots, N$. Although the problem and the method developed are general, we focus on the case $d = 3$ using only Euclidean coordinates as input. A neighborhood subset within radius r from a point x_i is denoted $B_r(x_i) = \{x_j | d_E(x_i, x_j) \leq r\}$ where $d_E(x_i, x_j) \in \mathbb{R}$ is the Euclidean (embedding) distance between x_i and x_j. The cardinality of $B_r(x_i)$ is K. The corresponding geodesic distance set around x_i is called $G_r(x_i) = \{g_{ij} = d_G(x_i, x_j) | x_j \in B_r(x_i)\}$ where $d_G \in \mathbb{R}$ means the geodesic distance. Our goal is to learn a function $f : x_i \mapsto G_r(x_i)$ that maps each point to (an approximation of) the geodesic distance set $G_r(x_i)$ around it.

3.3.2 Overview of the method

We introduce GeoNet, a network trained to learn the function f defined above. It consists of an autoencoder with skip connections, followed by a multi-scale Geodesic Matching (GM) layer, leveraging latent space features $\{\psi(x_i)\} \subseteq \mathbb{R}^{3+C}$ of the point set. GeoNet is trained in a supervised manner using ground truth geodesic distances between points in the set χ. To demonstrate the applicability of learned deep geodesic-aware representations from GeoNet, we test our approach on typical tasks that require understandings of the underlying surface topology, including point cloud upsampling, surface normal estimation, mesh reconstruction, and non-rigid shape classification. To this end, we leverage the existing state-of-the-art network architectures designed for the aforementioned problems. Specifically, we choose PU-Net as the baseline network for point upsampling and PointNet++ for other tasks. The proposed geodesic fusion methods, called PU-Net fusion (PUF) and PointNet++ fusion (POF), integrate GeoNet with the baseline or backbone models to conduct geodesic-aware point set analysis.

3.3.3 Geodesic neighborhood estimation

As illustrated in Fig. 3.2, GeoNet consists of two modules: an autoencoder that extracts a feature vector $\psi(x_i)$ for each point $x_i \in \chi$ and a GM layer that acts as a learned geodesic kernel function for estimating $G_r(x_i)$ using the latent features.

Feature Extraction. We use a variant of PointNet++, which is a point set based hierarchical and multi-scale function, for feature extraction. It maps an input point set χ to a feature set $\{\varphi(x_i)|x_i \in \widetilde{\chi}\}$ where $\varphi(x_i) \in \mathbb{R}^{3+\widetilde{C}}$ is a concatenation of the xyz coordinates and the $\widetilde{C}$ dimensional embedding of x_i, and $\widetilde{\chi}$ is a sampled subset of χ by farthest-point sampling. To recover features $\{\psi(x_i)\}$ for the point cloud χ, we use a decoder with skip connections. The decoder consists of recursively applied tri-linear feature interpolators, shared fully connected (FC) layers, ReLU and Batch Normalization. The resulting $(N, 3+C)$ tensor is then fed into the GM layer for geodesic neighborhood estimation.

Geodesic Matching. We group the latent features $\psi(x_i)$ into neighborhood feature sets $F_{r_l}(x_i) = \{\psi(x_j)|x_j \in B_{r_l}(x_i)\}$, under multiple radius scales r_l. At each scale r_l we set a maximum number of neighborhood points K_l, and thus produce a tensor of dimension $(N, K_l, 3+C)$. The grouped features, together with the latent features, are sent to a geodesic matching module, where $\psi(x_i)$ is concatenated with $\psi(x_j)$ for every $x_j \in B_{r_l}(x_i)$. The resulting feature $\xi_{ij} \in \mathbb{R}^{3+2C}$ becomes the input to a set of shared FC layers with ReLU, Batch Normalization and Dropout. As demonstrated in [HLJ15], the multilayer perceptron (MLP) acts as a kernel function that maps ξ_{ij} to an approximation of the geodesic distance, $\hat{g}_{ij}$. Finally, the GM layer yields $G_{r_l}(x_i)$ for each point of the input point cloud χ. We use a multi-scale L_1 loss $L_{geo} = \sum_l L_{geo_l}$ to compare the ground truth geodesic distances to their estimates:

$$L_{geo_l} = \sum_{x_i \in \chi} \sum_{x_j \in B_{r_l}(x_i)} \frac{|g_{ij} - \hat{g}(x_i, x_j)|}{N K_l} \tag{3.1}$$

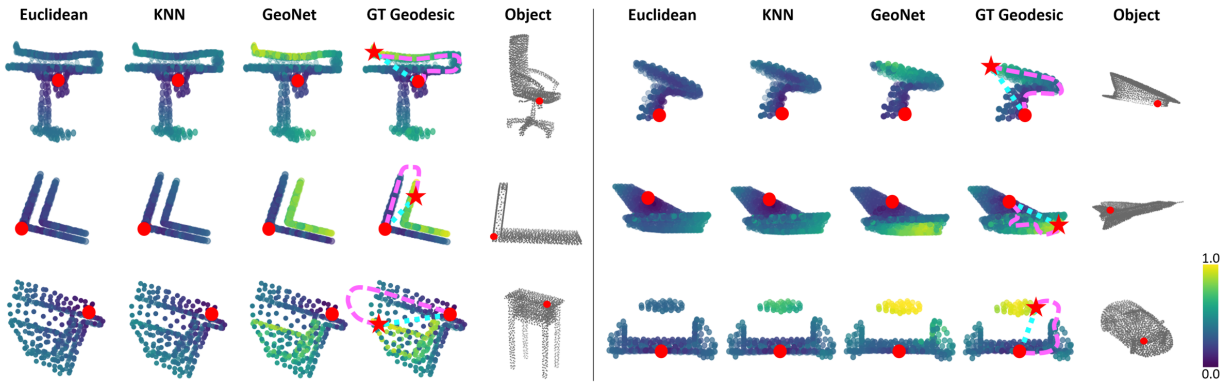


Figure 3.4: Representative results of geodesic neighborhood estimation. Red dots indicate the reference point and stars represent target points selected for the purpose of illustration. Points in dark-purple are closer to the reference point than those in bright-yellow. Shortest paths between the reference point and the target point in euclidean space are colored in sky-blue. Topology-based geodesic paths are in pink.

3.3.4 Geodesic fusion

To demonstrate how the learned geodesic representations can be used for point set analysis, we propose fusion methods based on the state-of-the-art (SOTA) network architectures for different tasks. For example, PU-Net is the SOTA upsampling method and thus we propose PUF that uses PU-Net as the baseline network to conduct geodesic fusion for point cloud upsampling. With connectivity information provided by the estimated geodesic neighborhoods, our geodesic-fused upsampling network can better recover topological details, such as curves and sharp structures, than PU-Net. We also present POF leveraging PointNet++ as the fusion backbone, and demonstrate its effectiveness on both geometric and semantic tasks where PointNet++ shows the state-of-the-art performance.

PU-Net Geodesic Fusion. A PUF layer, as illustrated in Fig. 3.3 (top), takes a (N, d) point set as input and sends it into two branches: one is a multi-scale Euclidean grouping layer, and the other is GeoNet. At each neighborhood scale r_l, the grouped point set $B_{r_l}(x_i)$ is fused with the estimated geodesic neighborhood $G_{r_l}(x_i)$ to yield $S_{r_l}(x_i) = \{(x_j, g_{ij}) | x_j \in B_{r_l}(x_i)\}$

with $(x_j, g_{ij}) \in \mathbb{R}^{d+1}$. Then the $(N, K_l, d+1)$ fused tensor is fed to a PointNet to generate a (N, C_l) feature tensor which will be stacked with features from other neighborhood scales. The remaining layers are from PU-Net. As indicated by the red rectangles in Fig. 3.3, the total loss has two weighted terms:

$$L = L_{geo} + \lambda L_{task} \tag{3.2}$$

where L_{geo} is for GeoNet training Eq. (3.1), λ is a weight and L_{task}, in general, is the loss for the current task that we are targeting. In this case, the goal is point cloud upsampling: $L_{task} = L_{up}(\theta)$ where θ indicates network parameters. PUF upsampling takes a randomly distributed sparse point set χ as input and generates a uniformly distributed dense point cloud $\hat{P} \subseteq \mathbb{R}^3$. The upsampling factor is $\alpha = \frac{|P|}{|\chi|}$:

$$L_{up}(\theta) = L_{EMD}(P, \hat{P}) + \lambda_1 L_{rep}(\hat{P}) + \lambda_2 \|\theta\|^2 \tag{3.3}$$

in which the first term is the Earth Mover Distance (EMD) between the upsampled point set $\hat{P}$ and the ground truth dense point cloud P:

$$L_{EMD}(P, \hat{P}) = \min_{\phi:\hat{P}\to P} \sum_{p_i \in \hat{P}} \|p_i - \phi(p_i)\|^2 \tag{3.4}$$

where $\phi : \hat{P} \to P$ indicates a bijection mapping.

The second term in Eq. (3.3) is a repulsion loss which promotes a uniform spatial distribution for $\hat{P}$ by penalizing close point pairs:

$$L_{rep}(\hat{P}) = \sum_{p_i \in \hat{P}} \sum_{p_j \in \widetilde{P}_i} \eta(\|p_i - p_j\|)\omega(\|p_i - p_j\|) \tag{3.5}$$

where $\widetilde{P}_i$ is a set of k-nearest neighbors of p_i, $\eta(r) = -r$ penalizes close pairs (p_i, p_j), and $\omega(r) = e^{-r^2/h^2}$ is a fast-decaying weight function with some constant h [HLZ09, LCL07].

PointNet++ Geodesic Fusion. Fig. 3.3 (bottom) illustrates the PointNet++ based fusion pipeline. Due to task as well as architecture differences between PU-Net and PointNet++, we make following changes to PUF to design a suitable fusion strategy that leverages PointNet++. First, for multi-scale grouping, we use the learned geodesic neighborhoods $\hat{G}_r(x_i)$ instead of Euclidean ones. Geodesic grouping brings attention to the underlying surfaces as well as structures of the point cloud. Second, while the PUF layer fuses estimated $\hat{G}_r(x_i) = \{\hat{g}_{ij} = \hat{d}_G(x_i, x_j) | x_j \in B_r(x_i)\}$, where $\hat{g}_{ij} \in \mathbb{R}$, of each neighborhood point set $B_r(x_i)$ into the backbone network, the POF layer uses the latent geodesic-aware features $\widetilde{\xi}_{ij} \in \mathbb{R}^{\widetilde{C}}$ extracted from the second-to-last FC layer in GeoNet. Namely, $\widetilde{\xi}_{ij}$ is an intermediate high-dimensional feature vector from ξ_{ij} to $\hat{g}_{ij}$ via FC layers, and therefore it is better informed of the intrinsic point cloud topology. Third, in PointNet++ fusion we apply the POF layer in a hierarchical manner, leveraging farthest-point sampling. Thus, the learned features encode both local and global structural information of the point set. The total loss for POF also has two parts: One is for GeoNet training and the other is for the task-at-hand. We experiment on representative tasks that can benefit from understandings of the topological surface attributes. We use the L_1 error for point cloud normal estimation:

$$L_{normal} = \sum_{x_i \in \chi} \sum_{j=1}^{3} \frac{\left| n_i^{(j)} - \hat{n}(x_i)^{(j)} \right|}{3N} \tag{3.6}$$

in which $n_i \in \mathbb{R}^3$ is the ground truth unit normal vector of x_i, and $\hat{n}(x_i) \in \mathbb{R}^3$ is the estimated normal. We then use the normal estimation to generate mesh via Poisson surface reconstruction [KH13a]. To classify point clouds of non-rigid objects, we use cross-entropy loss:

$$L_{cls} = -\sum_{c=1}^{S} y_c log(p_c(\chi)) \tag{3.7}$$

where S is the number of non-rigid object categories, and c is class label; $y_c \in \{0, 1\}$ is a binary indicator, which takes value 1 if class label c is ground truth for the input point set. $p_c(\chi) \in \mathbb{R}$ is the predicted probability w.r.t. class c of the input point set.

		K-3	*K-6*	*K-12*	*Euc*	GeoNet
v1	r $\leqslant$ 0.1	8.75	8.97	9.04	9.06	**5.67**
	r $\leqslant$ 0.2	16.22	17.33	17.90	18.16	**9.25**
	r $\leqslant$ 0.4	15.15	16.80	17.88	18.95	**9.75**
v2	r $\leqslant$ 0.1	11.71	11.49	11.55	11.57	**7.06**
	r $\leqslant$ 0.2	19.22	17.76	18.28	18.56	**9.74**
	r $\leqslant$ 0.4	21.03	17.19	18.20	19.44	**10.04**
v3	r $\leqslant$ 0.1	13.28	14.23	14.62	14.78	**10.86**
	r $\leqslant$ 0.2	14.85	17.27	18.54	19.49	**13.61**
	r $\leqslant$ 0.4	**13.48**	16.10	17.72	19.68	14.73

Table 3.1: Neighborhood geodesic distance estimation MSE (x100) on the heldout ShapeNet training-category samples. We compare with *KNN-Graph* based shortest path methods under different choices of K values. *Euc* represents the difference between Euclidean distance and geodesic distance. MSE(s) are reported under multiple radius ranges r. **v1** takes uniformly distributed point sets with 512 points as input, and **v2** uses randomly distributed point clouds. **v3** is tested using point clouds that have 2048 uniformly distributed points.

3.4 Implementation Details

For GeoNet training, the multiscale loss L_{geo_l} is enforced at three radius ranges: 0.1, 0.2 and 0.4. We use Adam [KB14] with learning rate 0.001 and batchsize 3 for 8 epochs. To train the geodesic fusion networks, we set the task term weight λ as 1, and use Adam with learning rate 0.0001 and batchsize 2 for around 300 to 1500 epochs depending on the task and the dataset.

3.5 Experiments

We put GeoNet to the test by estimating point cloud geodesic neighborhoods. To demonstrate the applicability of learned deep geodesic-aware representations, we also conduct experiments

		K-3	*K-6*	*K-12*	*Euc*	GeoNet
v1	r $\leqslant$ 0.1	8.81	9.01	9.05	9.06	**7.52**
	r $\leqslant$ 0.2	11.84	12.88	13.49	13.75	**11.44**
v2	r $\leqslant$ 0.1	10.52	10.21	10.25	10.26	**8.94**
	r $\leqslant$ 0.2	15.02	12.99	13.59	13.86	**11.69**
v3	r $\leqslant$ 0.1	11.82	12.39	12.65	12.75	**10.88**
	r $\leqslant$ 0.2	**11.80**	12.84	13.55	14.50	12.26

Table 3.2: Geodesic neighborhood estimation MSE (x100) on the leftout ShapeNet categories. **v1** takes uniformly distributed point sets with 512 points as input, and **v2** uses randomly distributed point clouds. **v3** is tested using point clouds that have 2048 uniformly distributed points.

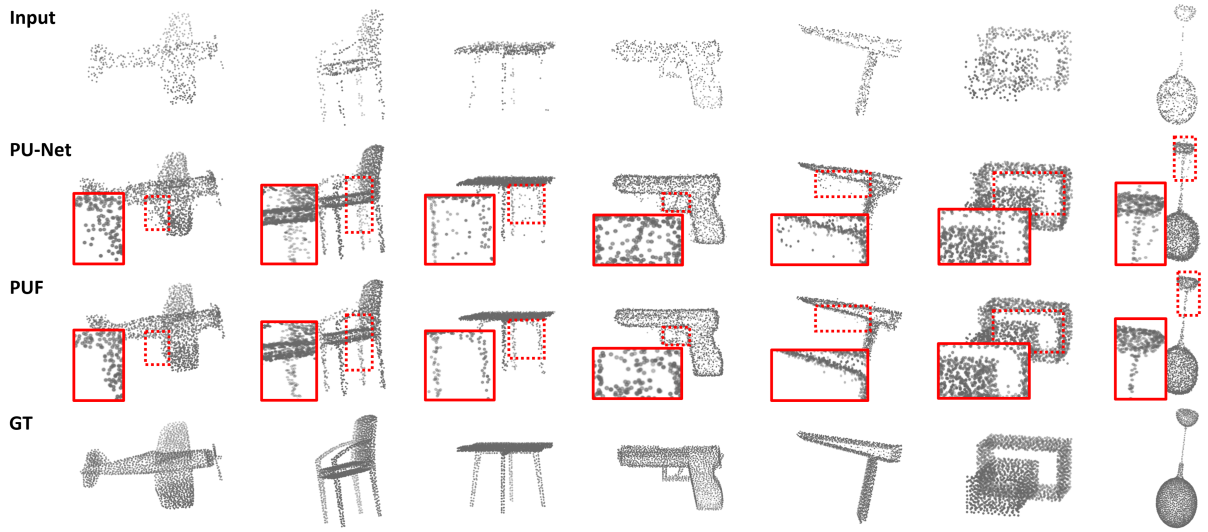


Figure 3.5: Point cloud upsampling comparisons with PU-Net. The input point clouds have 512 points with random distributions and the upsampled point clouds have 2048 points. Red insets show details of the corresponding dashed region in the reconstruction.

on down-stream point cloud tasks such as point upsampling, normal estimation, mesh reconstruction and non-rigid shape classification.

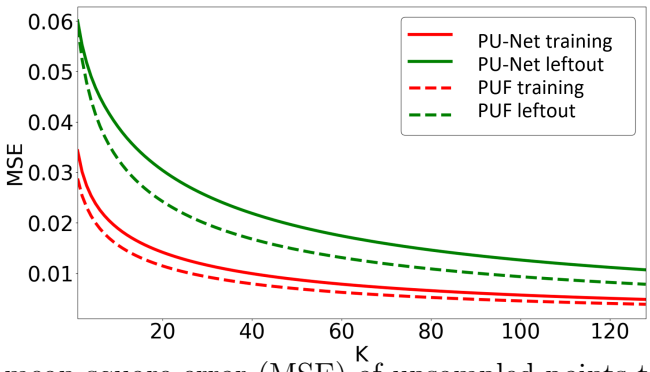


Figure 3.6: Top-k mean square error (MSE) of upsampled points that have large errors, for the heldout training-category samples (red) and the leftout ShapeNet categories (green).

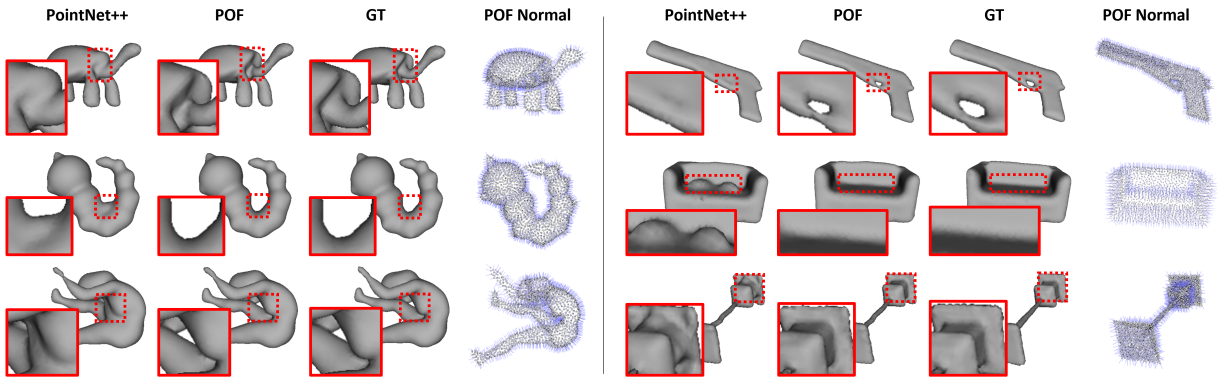


Figure 3.7: Mesh reconstruction results on the Shrec15 (left) and the ShapeNet (right) datasets using the estimated normal by PointNet++ and our method POF. GT presents mesh reconstructed via the ground truth normal. We also visualize POF normal estimation in the fourth and the last columns.

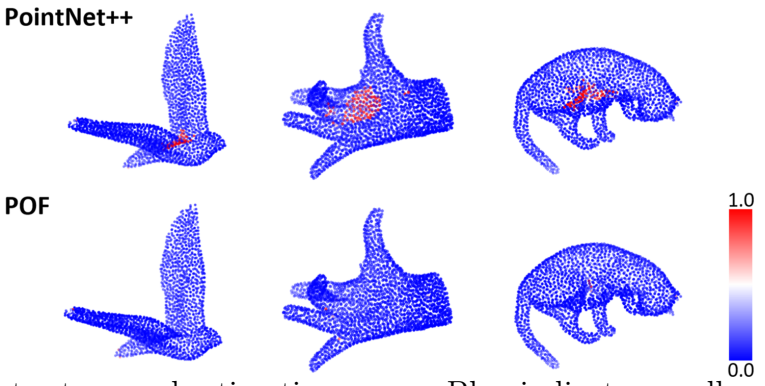


Figure 3.8: Point set normal estimation errors. Blue indicates small errors and red is for large ones.

		MSE	EMD	CD
Training	PU-Net	7.14	8.06	2.72
	PUF	**6.23**	**7.62**	**2.46**
Leftout	PU-Net	12.38	11.43	3.98
	PUF	**9.55**	**8.90**	**3.27**

Table 3.3: Point cloud upsampling results on both the heldout training-category samples and the unseen ShapeNet categories. MSE(s) (x10000) are scaled for better visualization.

3.5.1 Geodesic neighborhood estimation

In Tab. 3.1 (**v1**) we show geodesic distance set, $G_r(x_i)$, estimation results on the ShapeNet dataset [CFG15a] using point clouds with 512 uniformly distributed points. Mean-squared errors (MSE) are reported under multiple radius scales r w.r.t. $x_i \in \chi$. GeoNet demonstrates consistent improvement over the baselines. Representative results are visualized in Fig. 3.4. Our method captures various topological patterns, such as curved surfaces, layered structures, inner/outer parts, etc.

Generality. We test GeoNet's robustness under different point set distributions and sizes. In Tab. 3.1 (**v2**) we use point clouds with 512 randomly distributed points as input. We also test on dense point sets that contain 2048 uniformly distributed points in Tab. 3.1 (**v3**). Our results are robust to different point set distributions as well as sizes. To show the generalization performance, in Tab. 3.2 we report results on the leftout ShapeNet categories. Our method performs better on unseen categories, while *KNN-Graph* based shortest path approaches suffer from point set distribution randomness, density changes and unsuitable choices of K values.

3.5.2 Point cloud upsampling

We test PUF on point cloud upsampling and present results in Tab. 3.3. We compare against the state-of-the-art point set upsampling method PU-Net on three metrics: MSE, EMD as

	$\leqslant 2.5°$	$\leqslant 5°$	$\leqslant 10°$	$\leqslant 15°$
PCA	6.16±0.01	14.85±0.02	27.16±0.17	34.17±0.28
PointNet++	12.81±0.18	33.37±0.92	61.58±2.02	75.49±1.95
POF	**16.26**±0.30	**39.02**±1.09	**66.98**±1.46	**79.66**±1.21

Table 3.4: Point cloud normal estimation accuracy (%) on the Shrec15 dataset under multiple angle thresholds.

		$\leqslant 2.5°$	$\leqslant 5°$	$\leqslant 10°$	$\leqslant 15°$
Training	PCA	5.33	10.11	18.52	24.82
	PointNet++	30.68	43.19	55.91	62.30
	POF	**32.04**	**45.02**	**57.52**	**63.62**
Leftout	PCA	5.24	10.59	18.99	25.17
	PointNet++	17.35	28.82	43.26	51.17
	POF	**19.13**	**31.83**	**46.22**	**53.78**

Table 3.5: Point cloud normal estimation accuracy (%) on the ShapeNet dataset for both heldout training-category samples and leftout categories.

well as the Chamfer Distance (CD). Our method outperforms the baseline under all metrics by 9.25% average improvement on the heldout training-category samples. Since geodesic neighborhoods are better informed of the underlying point set topology than Euclidean ones, PUF upsampling produces less outliers and recovers more details in Fig. 3.5, such as curves and sharp structures.

Generality. To analyze outlier robustness (i.e. points with large reconstruction errors), we plot top-k MSE in Fig. 3.6. Our method generates fewer outliers on both the heldout training-category samples and the unseen categories. We also report quantitative results on the leftout categories in Tab. 3.3. Again, PUF significantly surpasses the state-of-the-art upsampling method PU-Net under three different evaluation metrics.

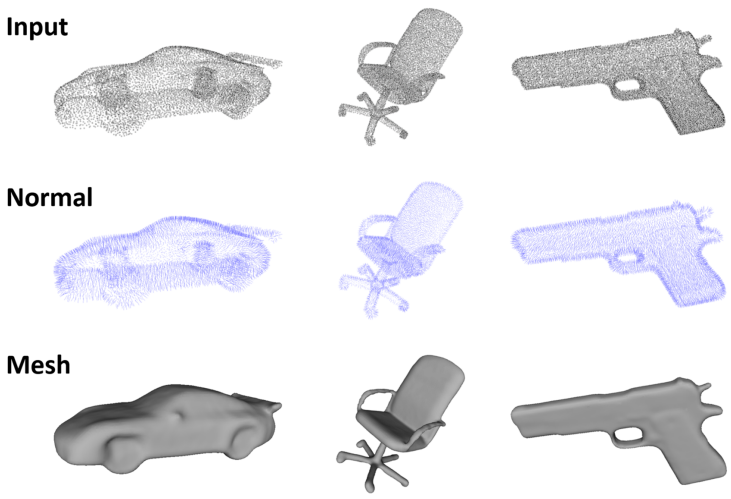


Figure 3.9: Normal estimation and Poisson mesh reconstruction results by POF using dense point clouds with 8192 points.

3.5.3 Normal estimation and mesh reconstruction

For normal estimation we apply PointNet++ geodesic fusion, POF, then we conduct Poisson mesh reconstruction leveraging the estimated normals. Quantitative results for normal estimation on the Shrec15 dataset and the ShapeNet dataset are given in Tab. 3.4 and Tab. 3.5, respectively. We compare our method with the traditional PCA algorithm as well as the state-of-the-art deep learning method PointNet++. Our results outperform the baselines by around 10% relative improvement. In Fig. 3.8, we visualize typical normal estimation errors, showing that PointNet++ usually fails at high-curvature and complex-surface regions. For further evidence, we visualize Poisson mesh reconstruction in Fig. 3.7 using the estimated normals.

Generality. In Tab. 3.5 we evaluate normal estimation performance on the leftout ShapeNet categories. Our method has higher accuracy over competing methods under multiple angle thresholds. Though trained with point clouds of 2048 points, POF is also tested on denser input. In Fig. 3.9 we take point clouds with 8192 points as input, and visualize the normal estimation and mesh reconstruction results, which shows that our method generalizes to dense point clouds without re-training and produces fine-scaled mesh.

	Input feature	Accuracy (%)
PointNet++	XYZ	73.56
POF	XYZ	**94.67**
DeepGM	Intrinsic features	93.03

Table 3.6: Point cloud classification of non-rigid shapes on the Shrec15 dataset.

	Gaussian Noise Level				
	0.8%	0.9%	1.0%	1.1%	1.2%
PointNet++	70.54	69.27	67.83	65.66	62.38
POF	91.89	90.93	89.40	87.72	84.98

Table 3.7: Noisy point clouds classification accuracy (%). We add Gaussian noise of 0.8% to 1.2% of unit ball radius.

3.5.4 Non-rigid shape classification

Results of non-rigid shape classification are reported in Tab. 3.6. While POF and PointNet++ only take point cloud-based xyz Euclidean coordinates as input, DeepGM requires offline computed intrinsic features from mesh data in the ground truth geodesic metric space. Though using less informative data, our method has higher classification accuracy than other methods, which further demonstrates that the proposed geodesic fusion architecture, POF, is suitable for solving tasks that require understandings of the underlying point cloud surface attributes.

Generality. We add Gaussian noise of different levels to the input and conduct noisy point clouds classification. Comparisons are shown in Tab. 3.7. POF outperforms PointNet++ under several noise levels. Our method also demonstrates better noise robustness. It shows a 10.24% decrease in relative accuracy at the maximum noise level, while PointNet++ decreases by up to 15.20%.

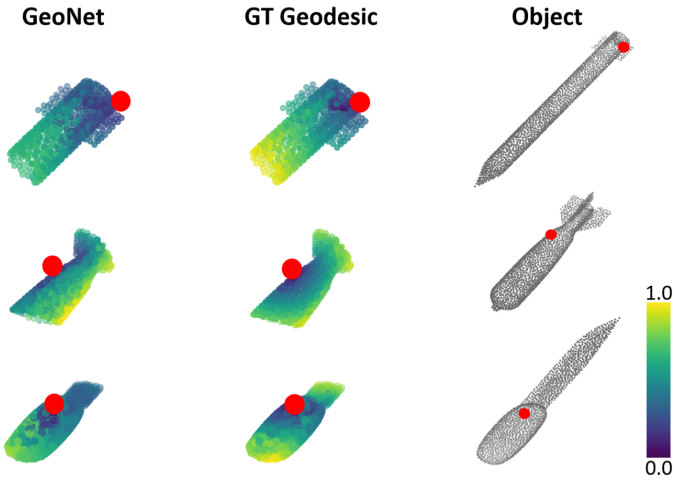


Figure 3.10: Failure cases of geodesic neighborhood estimation for stick-shaped objects (e.g. rocket, knife, etc.) which have large ratios between length and width/height. Red dots indicate the reference point. Points in dark-purple are closer to the reference point than those in bright-yellow.

3.5.5 Failure modes

Failure cases of geodesic neighborhood estimation are shown in Fig. 3.10. Due to large ratios between length and width/height, after normalizing a stick-shaped object (e.g. rocket, knife, etc.) into a unit ball we need high precision small values to represent its point-pair geodesic distance along the width/height sides. Since stick-shaped objects like rocket and knife only take up a small portion of the training data, GeoNet tends to make mistakes for heldout samples from these categories at inference time. Using an anistropic normalization might alleviate this issue but is challenging in practice as the principal directions would have to be estimated. We have not found additional failure cases, and quantitative improvements continue to take effect due to rich surface-based topological information learned during the geodesic-supervised training process.

3.6 Discussion

We have presented GeoNet, a novel deep learning architecture to learn the geodesic space-based topological structure of point clouds. The training process is supervised by the ground truth geodesic distance and therefore the learned representations reflect the intrinsic structure of the underlying point set surfaces. To demonstrate the applicability of such a topology estimation network, we also propose fusion methods to incorporate GeoNet into computational schemes that involve the standard backbone architectures for point cloud analysis. Our method is tested on both geometric and semantic tasks and outperforms the state-of-the-art methods, including point upsampling, normal estimation, mesh reconstruction and non-rigid shape classification. For future works we will move on to complicated scenes where an approximated mesh might need to be computed first in order to conduct the geodesic distance supervised training for GeoNet.

3.7 Additional Results

More results on geodesic neighborhood estimation, point cloud upsampling, normal estimation and mesh reconstruction are shown in Fig. 3.11, Fig. 3.12, Fig. 3.13 and Fig. 3.14.

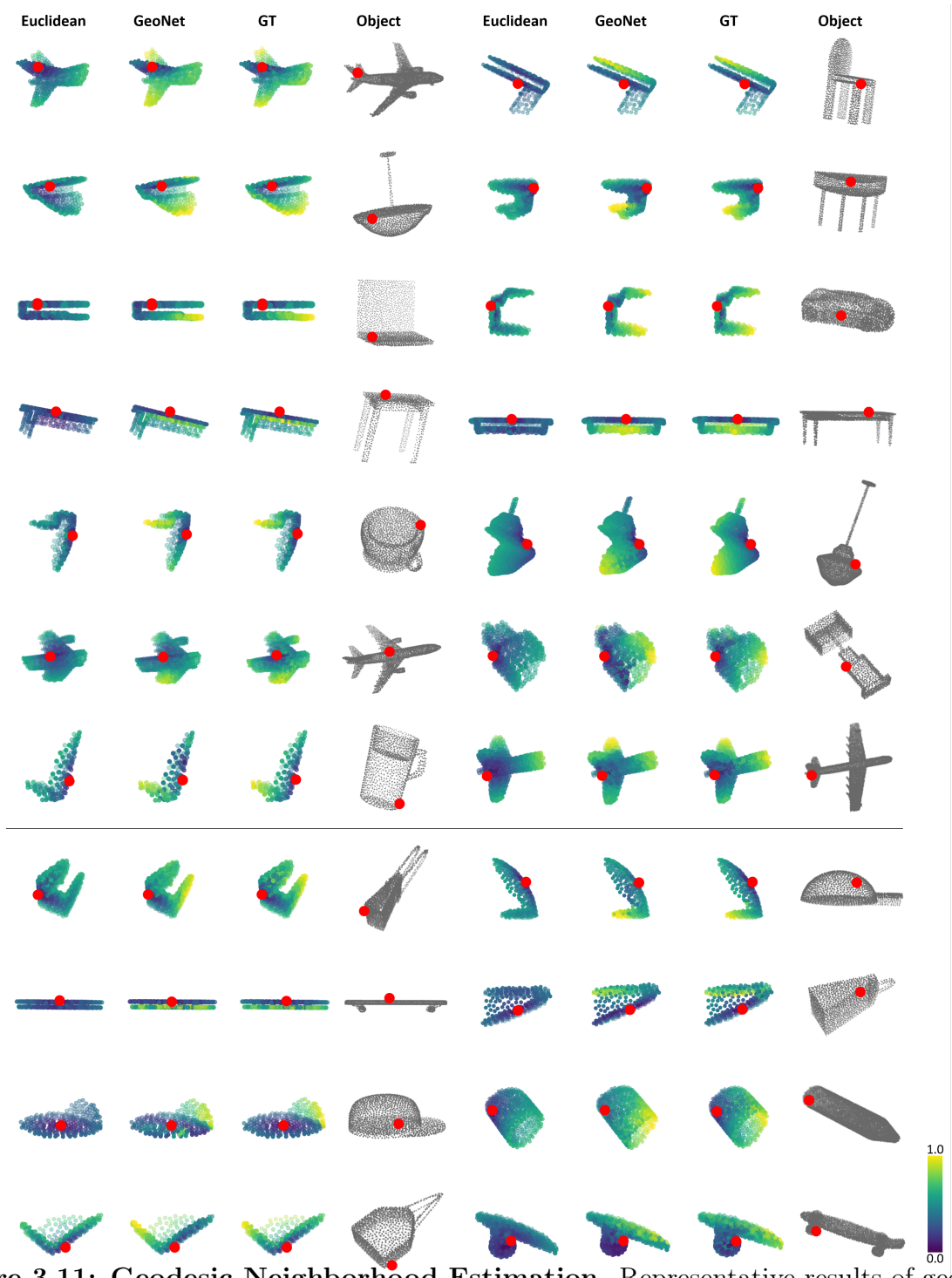


Figure 3.11: Geodesic Neighborhood Estimation. Representative results of geodesic neighborhood estimation using point clouds with 2048 points of uniform distribution. Red dots indicate the reference point. Points in dark-purple are closer to the reference point than those in bright-yellow. Held-out samples from the ShapeNet training categories are above the line and leftout categories are below the line.

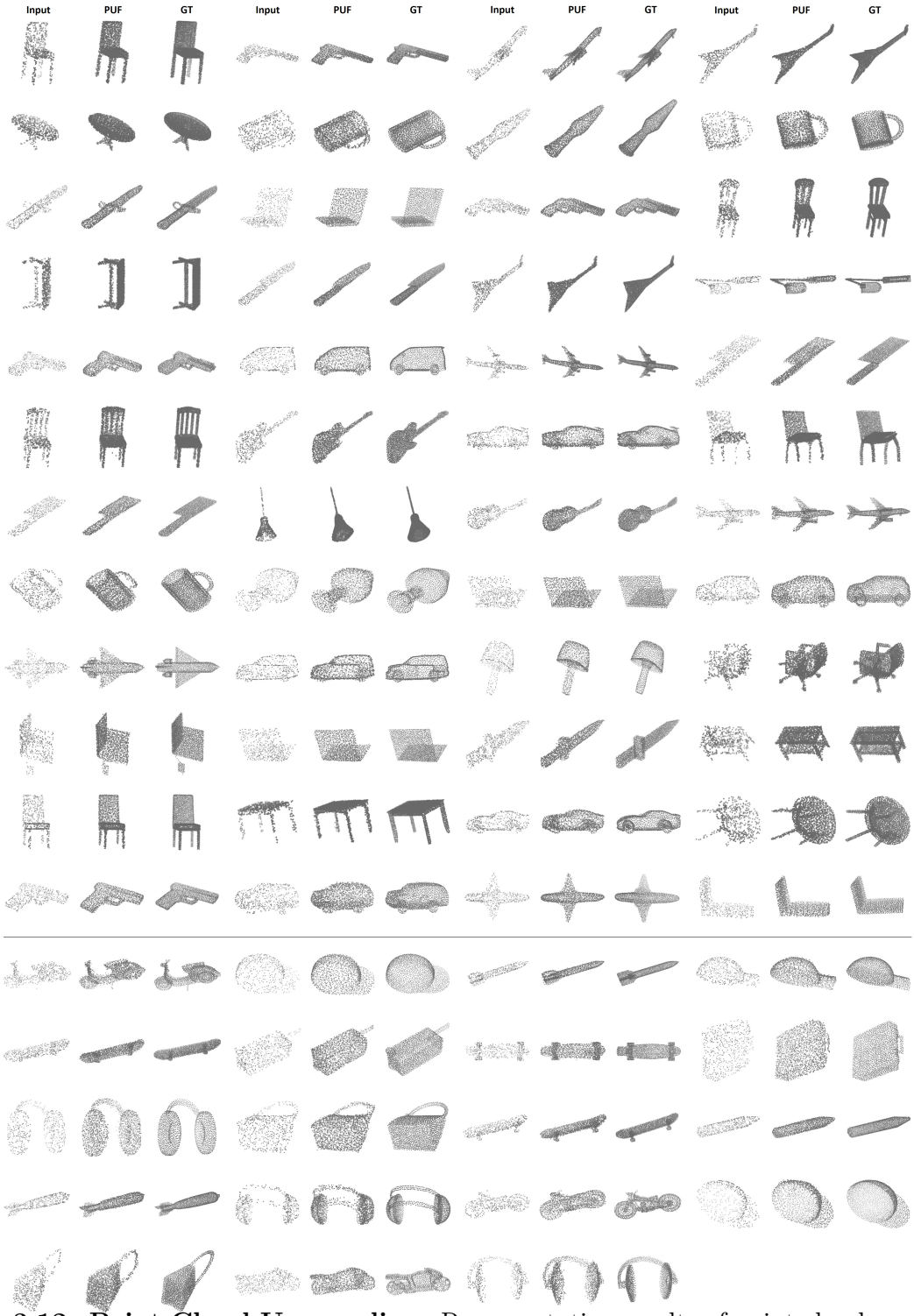


Figure 3.12: Point Cloud Upsampling. Representative results of point cloud upsampling. The input point clouds have 512 points with random distributions and the upsampled point clouds by our method, PUF, have 2048 points. Held-out samples from the ShapeNet training categories are above the line and leftout categories are below the line.

Figure 3.13: Normal Estimation and Mesh Reconstruction. Representative results of point cloud normal estimation and Poisson mesh reconstruction on the ShapeNet dataset. We test on input point clouds that have 8192 points though our method, POF, is trained using point clouds with 2048 points. Held-out samples from the ShapeNet training categories are above the line and leftout categories are below the line.

Figure 3.14: Normal Estimation and Mesh Reconstruction. Representative results of point cloud normal estimation and Poisson mesh reconstruction on the Shrec15 dataset. We test on input point clouds that have 8192 points though our method, POF, is trained using point clouds with 2048 points.

CHAPTER 4

Single Image-based 3D Shape Reconstruction

4.1 Introduction

Image based modeling is enabling new forms of immersive content production, particularly through realistic capture of human performance. Recently, deep implicit modeling techniques delivered a step change in 3D reconstruction of clothed human meshes from monocular images. These implicit methods train deep neural networks to estimate dense, continuous occupancy fields from which meshes may be reconstructed e.g. via Marching Cubes [LC87b].

Reconstruction from a single view is an inherently under-constrained problem. The resulting ambiguities are resolved by introducing assumptions into the design of the implicit surface function; the learned function responsible for querying a 3D point's occupancy by leveraging feature evidence from the input image. Prior work extracts either a global image feature [MON19b, PFS19b, CZ19a, LZP19], or pixel-aligned features (PIFu [SHN19b]) to drive this estimation. Neither approach takes into account fine-grain local shape patterns, nor seek to enforce global consistency to encourage physically plausible shapes and poses in the reconstructed mesh. This can lead to unnatural body shapes or poses, and loss of high-frequency surface details within the reconstructed mesh.

This paper contributes Geo-PIFU: an extension of pixel-aligned features to include three dimensional information estimated via a latent voxel representation that enriches the feature representation and regularizes the global shape of the estimated occupancy field. We augment the pixel-aligned features with *geometry-aligned shape features* extracted from a latent voxel

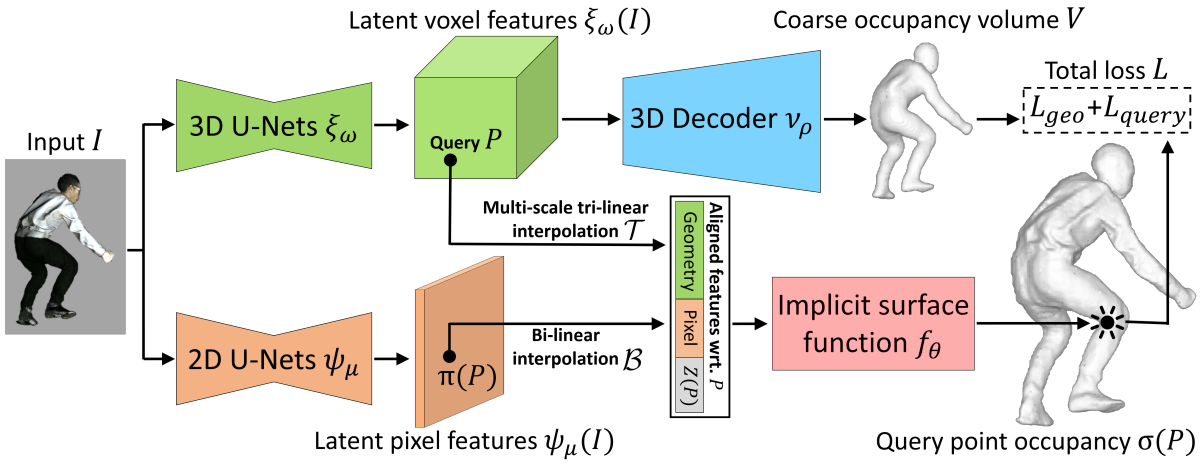


Figure 4.1: Pipeline. Our method extracts latent 3D voxel and 2D pixel features from a single-view color image. The extracted features then are used to compose geometry and pixel aligned features *wrt.* each query point P for occupancy estimation through an implicit surface function. At training time, we enforce losses on both the coarse occupancy volume and the estimated query point occupancy values. Note that the blue color 3D convolution decoder for generating the coarse occupancy volume is only needed at training time to supervise the latent voxel features.

feature representation obtained by lifting the input image features to 3D. These voxel features naturally align with the human mesh in 3D space and thus can be trilinearly interpolated to obtain a query point's encoding. The uniform coarse occupancy volume losses and the structure-aware 3D U-Nets architecture used to supervise and generate the latent voxel features, respectively, both help to inject global shape topology robustness / regularities into the voxel features. Essentially, the latent voxel features serve as a coarse human shape proxy to constrain the reconstruction. Figure 7.2 summarises our proposed adaptation (see upper branch) to reconstruct a clothed human mesh from a single-view image input using a geometry and pixel aligned implicit surface function. Our three technical contributions are:

1. Geometry and pixel aligned features. We fuse geometry (3D, upper branch) and pixel (2D, lower branch) aligned features to resolve local feature ambiguity. For instance, query

points lying upon similar camera rays but on the front or back side of an object are reprojected to similar pixel coordinates resulting in similar interpolated 2D features. Incorporating geometry-aligned shape features derived from our latent voxel feature representation resolves this ambiguity, leading to clothed human mesh reconstructions with rich surface details.

2. Global shape proxy. We leverage the same latent voxel feature representation as a coarse human shape proxy to regularize reconstruction and encourage plausible global human shapes and poses; crucial given the requirement to hallucinate the unobserved rear of the imaged object. This improves the plausibility of the estimated character shape and pose with accurate ground truth 3D alignment, and reduces mesh artifacts like distorted hands and feet.

3. Scale of Evaluation. We extend the evaluation of deep implicit surface methods to the DeepHuman dataset [ZYW19a]: 5436 training meshes, 1359 test meshes — 10 times larger then the private commercial dataset used in PIFu. The leading results on this dataset are generated by voxel and parametric mesh representations based methods. No deep implicit function method has been benchmarked on it before.

We show that Geo-PIFu exceeds the state of the art deep implicit function method PIFu [SHN19b] by 42.7% reduction in Chamfer and Point-to-Surface Distances, and 19.4% reduction in normal estimation errors, corresponding to qualitative improvements in both local surface details and global mesh topology regularities.

4.2 Related Work

Our method is most directly related to the use of deep implicit function representations for single-view mesh reconstruction. For context, we will also briefly review on explicit shape representations.

4.2.1 Implicit Surface Function

Occupancy Networks [MON19b], DeepSDF [PFS19b], LIF [CZ19a] and DIST [LZP19] proposed to use global representations of a single-view image input to learn deep implicit surface functions for mesh reconstruction. They demonstrated state-of-the-art single-view mesh reconstruction results on the ShapeNet dataset [CFG15b] of rigid objects with mostly flat surfaces. However, the global representation based implicit function does not have dedicated query point encodings, and thus lacks modeling power for articulated parts and fine-scale surface details. This motivates later works of PIFu [SHN19b] and DISN [XWC19]. They utilize pixel-aligned 2D local features to encode each query point when estimating its occupancy value. The alignment is based on (weak) perspective projection from query points to the image plane, followed by bilinear image feature interpolation. PIFu for the first time demonstrated high-quality single-view mesh reconstruction for clothed human with rich surface details, such as clothes wrinkles. However, PIFu still suffers from the feature ambiguity problem and lacks global shape robustness. Another two PIFu variations are PIFuHD [SSS20] and ARCH [HXL20]. PIFuHD leverages higher resolution input then PIFu through patch-based feature extraction to accommodate GPU memory constraints. ARCH combines parametric human meshes (*e.g.* SMPL [LMR15]) with implicit surface functions in order to assign skinning weights for the reconstructed mesh and enable animations. Both methods require more input / annotations (e.g. 2× higher resolution color images, SMPL registrations) than PIFu. Note that PIFuHD, ARCH and Geo-PIFu focus on different aspects and are complementary. If provided with those additional data, our method can also integrate their techniques to further improve the performance. Another related work that also utilizes latent voxel features for implicit function learning is IF-Net [CAP20], but its problem setting is different from ours. While IF-Net takes partial or noisy 3D voxels as input, (Geo)-PIFu(HD) and ARCH only utilize a single-view color image. Thus, IF-Net has access to "free" 3D shape cues of the human subject. But Geo-PIFu must achieve an ill-posed 2D to 3D learning problem. Meanwhile, Geo-PIFu needs to factorize out pixel domain nuisances (*e.g.* colors, lighting) in

order to robustly recover the underlying dense and continuous occupancy fields.

4.2.2 Explicit Shape Models

Explicit shape models can be classified by the type of 3D shape representation that they use: voxel, point cloud or (parametric) mesh [VCR18a, JMT18a, LKL18, FSG17, YHH19, HHY19, GFK18, WZL18, RHP17, LMR15, KBJ18a, ZYW19a, APT19a]. Here we only review some representative works on single-view clothed human body modeling and reconstruction. BodyNet [VCR18a] leverages intermediate tasks (*e.g.* segmentation, 2D/3D skeletons) and estimates low-resolution body voxel grids. A similar work to BodyNet is VRN [JMT18a], but it directly estimates occupancy voxels without solving any intermediate task. Voxel-based methods usually are constrained by low resolution and therefore struggle to recover shape details. Driven by data efficiency and topology modeling flexibility of point clouds, several methods [LKL18, FSG17, YHH19] propose to estimate point coordinates from the input image. The drawback is that generating a mesh with fine-scale surface details by Poisson reconstruction [KBH06] requires a huge number of point estimations. In order to directly generate meshes, Atlas-Net [GFK18] uses deep networks to deform and stitch back multiple flat-shape 2D grids. However, the reconstructed meshes are not water-tight because of the stitching hole artifacts. Pixel2Mesh [WZL18] generates water-tight meshes by progressively deforming a sphere template. But the empirical results show that mesh-based methods produce overly smooth surfaces with limited surface details due to shape flexibility constraints of the template. Parametric shape models like SMPL [LMR15], BlendSCAPE [HLR12], *etc* are also useful for directly generating human meshes. For example, [KBJ18a, PZZ18, KPD19a, KPB19a, PKD19] estimate or fit the shape and pose coefficients of a SMPL model from the input image. Note that these parametric human body models are usually designed to be “naked", ignoring clothes shapes. Recently, methods that leverage hybrid explicit models are also proposed to combine the benefits of different shape representations. DeepHuman [ZYW19a] and Tex2Shape [APT19a] combine SMPL with methods of voxel refinement and

mesh deformation, respectively, for single-view clothed human mesh reconstruction. One issue of using parametric body shapes is non-trivial data annotation (*e.g.* registering SMPL models with ground-truth clothed human meshes), which is not needed in deep implicit surface function-based methods. Another problem is propagation error: wrong SMPL estimation can cause additional errors in its afterward steps of local surface refinement.

4.3 Methodology

Continuous occupancy fields map each 3D point inside it to a positive or negative occupancy value dependent on the point is inside the target mesh or not. We encode surface as the occupied / unoccupied space decision boundary of continuous occupancy fields and apply the Marching Cube algorithm [LC87b] to recover human meshes. Occupancy values are denoted by $\sigma \in \mathbb{R}$. Points inside a mesh have $\sigma > 0$ and outside are $\sigma < 0$. Thus, the underlying surface can be determined at $\sigma := 0$. In this work the goal is to learn a deep implicit function $f(\cdot)$ that takes a single-view color image $I : D \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$ and coordinates of any query point $P \in \mathbb{R}^3$ as input, *and* outputs the occupancy value σ of the query point: $f \mid (I, P) \mapsto \sigma$. The training data for learning the implicit surface function consists of (image, query point occupancy) pairs of $\{I, \sigma(P)\}$. More specifically, we densely sample training query points on the mesh and generate additional samples by perturbing them with Gaussian noise. In the following sections, we introduce the formulation of our deep implicit function in Section 4.3.1 and explain our training losses in Section 4.3.2. Implementation details are provided in Section 4.4.

4.3.1 Geo-PIFu

As shown in Figure 7.2, we propose to encode each query point P using fused geometry (3D, upper branch) and pixel (2D, lower branch) aligned features. Our method, Geo-PIFu, can be formulated as:

$$f_\theta(\mathcal{T}(\xi_\omega(I), P), \mathcal{B}(\psi_\mu(I), \pi(P)), Z(P))) := \sigma \tag{4.1}$$

The implicit surface function is denoted by $f_\theta(\cdot)$ and implemented as a multi-layer perceptron with weights θ. For each query point P, inputs to the implicit function for occupancy $\sigma(P)$ estimation include three parts: geometry-aligned 3D features $\mathcal{T}(\xi_\omega(I), P)$, pixel-aligned 2D features $\mathcal{B}(\psi_\mu(I), \pi(P))$, and the depth $Z(P) \in \mathbb{R}$ of P. They compose the aligned features *wrt.* P.

4.3.1.1 Geometry-aligned Features

First we explain geometry-aligned features $\mathcal{T}(\xi_\omega(I), P)$ in Equation (4.1). Here $\xi_\omega(\cdot)$ are 3D U-Nets [JBA17] that lift the input image I into 3D voxels of latent features. The networks are parameterized by weights ω. To extract geometry-aligned features from $\xi_\omega(I)$ *wrt.* a query point P, we conduct multi-scale trilinear interpolation $\mathcal{T}$ based on the xyz coordinates of P. This interpolation scheme is inspired by DeepVoxels [STH19a, HCJ20a] and IF-Net [CAP20]. It is important to notice that these voxel features naturally align with the human mesh in 3D space and thus can be trilinearly interpolated to obtain a query point's encoding. Specifically, we trilinearly interpolate and concatenate features from a point set Ω determined around P.

$$\Omega := \left\{P + n_i \mid n_i \in d \cdot \left\{(0,0,0)^T, (1,0,0)^T, (0,1,0)^T, (0,0,1)^T ...\right\}\right\} \tag{4.2}$$

Here $n_i \in \mathbb{R}^3$ is a translation vector defined along the axes of a Cartesian coordinate with step length $d \in \mathbb{R}$. For convenience, we will use trilinear interpolation to indicate our multi-scale feature interpolation scheme $\mathcal{T}$ for the rest of the paper. Not like global image features and pixel-aligned features, as discussed in Section 7.1, geometry-aligned features provide dedicated representations for query points that directly capture their local shape patterns and tackle the feature ambiguity problem. Since computation cost of latent voxel features

learning is usually a concerned issue, we emphasize that our latent voxel features are of low resolution (C-8, D-32, H-48, W-32), in total 393216. In comparison, the latent pixel features resolution is (C-256, H-128, W-128), in total 4194304. More discussion on the model parameter numbers is provided in the experiment section.

4.3.1.2 Pixel-aligned Features

Pixel-aligned features are denoted as $\mathcal{B}(\psi_\mu(I), \pi(P))$ in Equation (4.1), where π represents (weak) perspective camera projection from the query point P to the image plane of I. Then pixel-aligned features can be extracted from the latent pixel features $\psi_\mu(I)$ by bilinear interpolation $\mathcal{B}$ at the projected pixel coordinate $\pi(P)$. Image features mapping function $\psi_\mu(\cdot)$ is implemented as 2D U-Nets with weights μ. While geometry-aligned features, by designs of network architectures and losses, are mainly informed of coarse human shapes, pixel-aligned features focus on extracting high-frequency shape information from the single-view input, such as clothes wrinkles. In Section 4.5.2 we conduct ablation studies on single-view clothed human mesh reconstruction using individual type of features. The results empirically demonstrated these claimed attributes of geometry and pixel aligned features.

4.3.2 Training Losses

Next we explain the losses used in training the geometry and pixel aligned features, as well as the deep implicit surface function.

4.3.2.1 Coarse occupancy volume loss

To learn latent voxel features $\xi_\omega(I) \in \mathbb{R}^{m \times s_1 \times s_2 \times s_3}$ that are informed of coarse human shapes, we use a 3D convolution decoder and a sigmoid function to estimate a low-resolution occupancy volume $\hat{V}(I) \in \mathbb{R}^{1 \times d \times h \times w}$. We use extended cross-entropy losses [JBA17] for training.

Figure 4.2: Left: the DeepHuman dataset contains various clothes and poses. Right: the dataset used in PIFu consists of mostly upstanding poses.

$$L_{geo} = -\frac{1}{|\hat{V}|}\sum_{i,j.k} \gamma V^{i,j.k} \log \hat{V}^{i,j,k} + (1-\gamma)(1-V^{i,j,k}) \log(1-\hat{V}^{i,j,k}), \text{ with } \hat{V}(I) = \nu_\rho(\xi_\omega(I)) \quad (4.3)$$

The ground truth coarse occupancy fields $V \in \mathbb{R}^{1\times d\times h\times w}$ are voxelized from the corresponding high-resolution clothed human mesh. $(i, j.k)$ are indices for the (depth, height, width) axes, and γ is a weight for balancing losses of grids inside / outside the mesh. $\nu_\rho(\cdot)$ represents the 3D convolution decoder and the sigmoid function used for decoding the latent voxel features $\xi_\omega(I)$. As shown in Figure 7.2, this 3D convolution decoder is only needed at training time to provide supervision for the latent voxel features. At inference time, we just maintain the latent voxel features for trilinearly interpolating geometry-aligned 3D features. Compared with learning dense continuous occupancy fields, this coarse occupancy volume estimation task is easier to achieve. The learned voxel features are robust at estimating shapes and poses for the visible side and also hallucinating plausible shapes and poses for the self-occluded invisible side. This is critical when the human has complex poses or large self-occlusion. Meanwhile, using structure-aware 3D U-Nets to generate these voxel-shape features also helps inject shape regularities into the learned latent voxel features.

4.3.2.2 High-resolution query point loss

The deep implicit surface function is learned by query point sampling based sparse training. It is difficult to directly estimate the full spectrum of dense continuous occupancy fields, and therefore, at each training step, we use a group of sparsely sampled query points (*e.g.* 5000 in PIFu and Geo-PIFu) to compute occupancy losses. As shown in Equation (4.1) as well as Figure 7.2, for each query point P the deep implicit function utilizes geometry and pixel aligned features to estimate its occupancy value $\hat{\sigma}$. We use mean square losses in training.

$$L_{query} = \frac{1}{Q}\sum_{q=1}^{Q}(\sigma_q - \hat{\sigma}_q)^2 \tag{4.4}$$

The number of sampled query points at each training step is Q and the index is q. The ground-truth occupancy value of a query point P is σ. The total losses used in Geo-PIFu are $L = L_{geo} + L_{query}$. Following [MGK19, PKK19] we adopt a staged training scheme of the coarse occupancy volume loss and the high-resolution query point loss. While L_{geo} is designed to solve a discretized coarse human shape estimation task, L_{query} utilizes sparse query samples with continuous xyz coordinates and learns high-resolution continuous occupancy fields of clothed human meshes. Experiments in Section 4.5.2 demonstrated that the query point loss is critical for learning high-frequency shape information from the input image, and enables the deep implicit surface function to reconstruct sharp surface details.

4.4 Implementation Details

We use PyTorch [PGM19] with RMSprop optimizer [TH12] and learning rate $1e-3$ for both losses. We stop the 3D decoder training after 30 epochs and the implicit function training after 45 epochs using AWS V100-6GPU. The learning rate is reduced by a factor of 10 at the $8th$, $23th$ and $40th$ epochs. We use a batch of 30 and 36 single-view images, respectively. For each single-view image, we sample $Q = 5000$ query points to compute L_{query}. The step

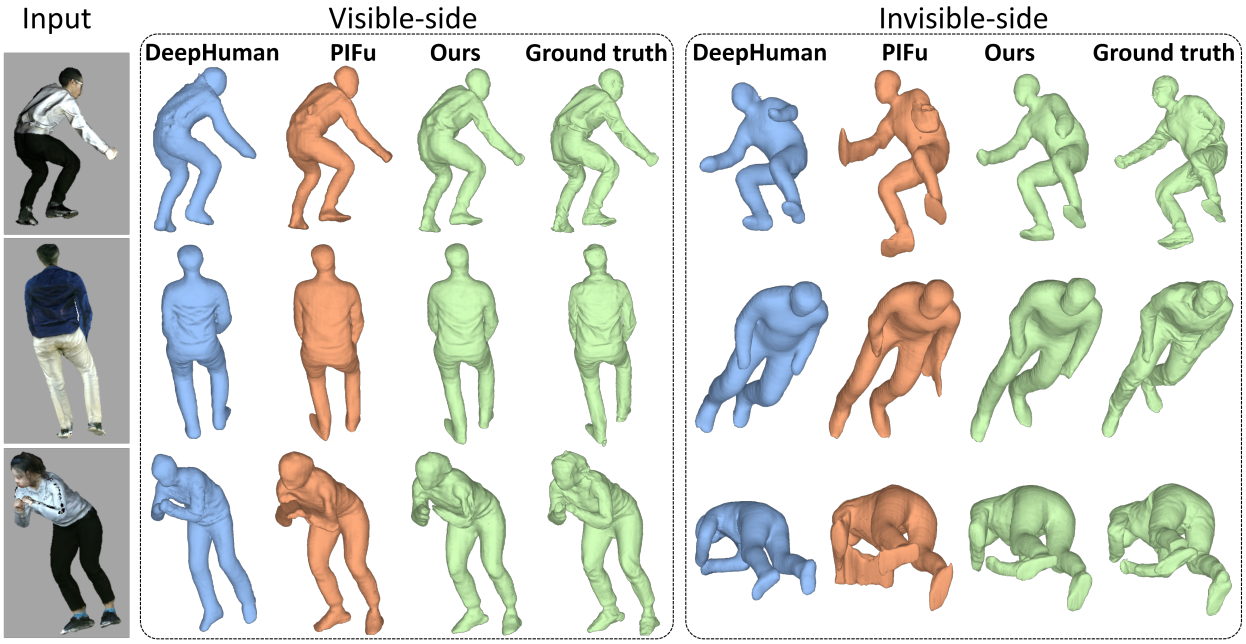


Figure 4.3: Single-view clothed human mesh reconstruction. Our results have less shape artifacts and distortions than PIFu. Besides improved global regularities and better alignment with ground truth, our meshes also contain more accurate and clear local surface details than DeepHuman and PIFu. DeepHuman can generate physically plausible topology benefiting from the voxelized SMPL model, but is incapable of capturing rich surface details due to limited voxel resolutions. Meanwhile, PIFu lacks global robustness when reconstructing meshes of complex poses and large self-occlusion.

length d for multi-scale tri-linear interpolation is 0.0722 and we stop collecting features at $2\times$ length. The balancing weight γ of L_{geo} is 0.7.

4.5 Experiments

We compare Geo-PIFu against other competing methods on single-view clothed human mesh reconstruction in Section 4.5.1. To understand how do the learned 3D-geometry and 2D-pixel aligned features impact human mesh reconstruction, we also show ablation studies on

individual type of features and different feature fusion architectures in Section 4.5.2. Failure cases and effect of 3D generative adversarial network (3D-GAN) training are discussed in Section 4.5.3. More qualitative results and network architecture details can be found in Section 4.7.

Dataset We use the DeepHuman dataset [ZYW19a]. It contains 5436 training and 1359 test human meshes of various clothes and poses. In comparison, the dataset used by PIFu [SHN19b] only has 442 training and 54 test meshes of mostly upstanding poses. Mesh examples are shown in Figure. 4.2. More importantly, our dataset is public and the meshes are reconstructed from cheap RGB-D sensors, while the PIFu dataset is private commercial data well-polished by artists. This means that our results can all be reproduced and the mesh collection procedures can be easily expanded to other domain-specific scenarios to obtain more human meshes (*e.g.* basketball players). Another critical difference is that we support free camera rotation when rendering the training and test images, as shown in the input column of Figure 4.3. In contrast, PIFu images are all rendered with zero elevation and in-plane rotation, namely the camera is always facing upright towards the mesh. In brief, our dataset is public and has more diversities in human subjects, poses, clothes and rendering angles. Therefore, our dataset is more challenging to learn and less likely to cause over-fitting on upstanding human poses and horizontal camera angles than the dataset used in PIFu. The downside of the DeepHuman dataset is that it lacks high quality texture maps for photo-realistic rendering, which might hurt model generalization on in-the-wild natural images.

Metrics To evaluate reconstructed human meshes, we compute Chamfer Distance (CD) and Point to Surface Distance (PSD) between the reconstructed mesh and the ground truth mesh. While CD and PSD focus more on measuring the global quality of the mesh topology, we also compute Cosine and *L2* distances upon the mesh normals to evaluate local surface details, such as clothes wrinkles. These metrics are also adopted in PIFu for human mesh reconstruction evaluation.

	Mesh		Normal	
	CD	PSD	Cosine	*L2*
DeepHuman	11.928	11.246	0.2088	0.4647
PIFu	2.604	4.026	0.0914	0.3009
Ours	**1.742**	**1.922**	**0.0682**	**0.2603**

Table 4.1: Benchmarks. These methods are all trained with the same DeepHuman dataset for fair comparisons. To evaluate global topology accuracy of meshes, we report CD ($\times$10000) and PSD ($\times$10000) between the reconstructed human mesh and the ground truth mesh. We also compute Cosine and *L2* distances for the input view normals to measure fine-scale surface details, such as clothes wrinkles. Small values indicate good performance. Our approach outperforms the competing methods, demonstrating the global / local advantages of utilizing geometry and pixel aligned features for deep implicit surface function learning.

Method	Parameter Size	Mesh		Normal	
		CD	PSD	Cosine	*L2*
PIFu	15604738	10.571	9.285	0.1422	0.4141
PIFuHD	387049625	9.489	9.349	0.1228	0.3776

Table 4.2: Benchmarks. These two models are evaluated using pre-trained weights provided by their authors. Parameter size of Geo-PIFu is 30616954 (*12 times smaller than PIFuHD*).

Competing Methods Recently DeepHuman [ZYW19a], PIFu [SHN19b] and PIFuHD [SSS20] have demonstrated SOTA results on single-view clothed human mesh reconstruction.

1) DeepHuman integrates the benefits of parametric mesh models and voxels. It first estimates and voxelizes a SMPL model [LMR15] from a color image, and then refines the voxels through 3D U-Nets. To improve surface reconstruction details, they further estimate a normal map from the projected voxels and use the estimated normals to update the human mesh. DeepHuman shows large improvement over both parametric shapes and voxel representations. Therefore we compare Geo-PIFu against it. Note that this is not a fair

comparison, because DeepHuman requires registered SMPL parametric meshes as additional data annotation.

2) PIFu leverages pixel-aligned image features to encode each query point for implicit surface function-based continuous occupancy fields learning. This method is most related to our work since we both use implicit surface function for 3D shape representation. Note that PIFu and Geo-PIFu have the same requirement of color image inputs and ground-truth training meshes.

3) PIFuHD is built upon PIFu with higher resolution inputs than all other competing methods and than our method. It also uses offline estimated front/back normal maps to further augment input color images. These additional modules make PIFuHD a parameter-heavy model (see Table 4.2).

4.5.1 Single-view Reconstruction Comparisons with State-of-the-art

Global Topology Results of CD / PSD between the recovered human mesh and the ground truth mesh are provided in the left column of Table 4.1. As explained in the metrics sections, these two metrics focus more on measuring the global quality of the mesh topology. Our results surpass the second best method PIFu by 42.7% relative error reduction on average. Such improvement indicates that human meshes recovered by our method have better global robustness / regularities and align better with the ground truth mesh. These quantitative findings as well as analysis have also been visually proved in Figure 4.3. Our mesh reconstructions have fewer topology artifacts and distortions (*e.g.* arms, feet) than PIFu, and align better with the ground truth mesh than DeepHuman. Moreover, we include results of pre-trained models released by PIFu and PIFuHD in Table 4.2, because the latter has not yet released its training scripts. Under the same training data, PIFuHD achieves lower relative improvement over PIFu than Geo-PIFu. Note that the two ideas of PIFuHD (using sliding windows to ingest high resolution images, and offline estimated front/back normal maps to further augment input color images) are both add-on modules *wrt.* (Geo)-PIFu.

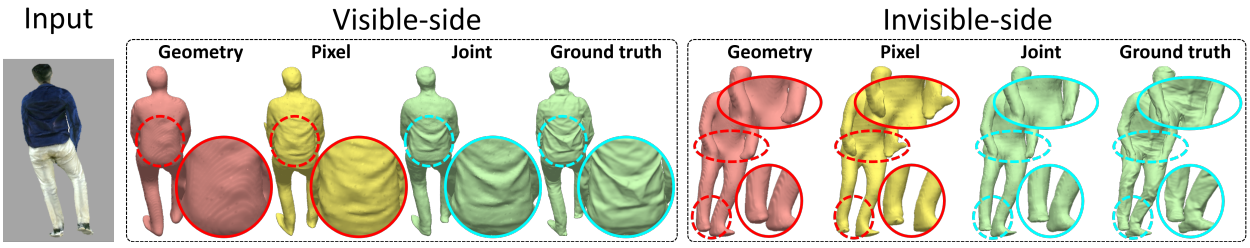


Figure 4.4: Left to right: mesh reconstruction results of exp-a, b, e in Table 4.3. This comparison shows that mesh reconstructions using only the 3D geometry features have better global topology regularities and ground truth alignment, but contain fewer local surface details than meshes reconstructed from only the 2D pixel features. The best global / local performances are achieved when two types of representations are jointly used for deep implicit surface function learning.

Given high resolution images and offline estimated normal maps, one might combine PIFuHD with Geo-PIFu for further improved local surface details and global topology robustness. But this is out of the scope of our work.

Local Details Besides improved global topology, our mesh reconstruction results contain more local surface details such as clothes wrinkles. The ability of capturing fine-scale surface details is quantitatively measured in the right column of Table 4.1. We compute Cosine and $L2$ distances between the input view normals of the reconstructed mesh and those of the ground truth mesh. Our approach improves over the second best method PIFu by 19.4% relative error reduction on average. To further explain the improved global and local performances of our method, we conduct ablation studies on individual type of features and different feature fusion architectures in the next section.

4.5.2 Ablation Studies

Geometry vs. Pixel Aligned Features In our method, the implicit function utilizes both the geometry-aligned 3D features and the pixel-aligned 2D features to compose query

	Mesh		Normal	
	CD	PSD	Cosine	*L2*
a. Ours-geometry	2.293	2.629	0.0864	0.3164
b. Ours-pixel	2.604	4.026	0.0914	0.3009
c. Ours-late-3D	1.753	1.930	**0.0675**	**0.2598**
d. Ours-late-both	**1.677**	**1.868**	0.0764	0.2767
e. Ours-early-3D	**1.742**	**1.922**	**0.0682**	**0.2603**

Table 4.3: Ablation studies. To analyze impact of the learned latent voxel, pixel features, we report human mesh reconstruction results that use individual type of features and the fused features. Small values indicate good performance. Exp-a, b evaluate meshes reconstructed from solely 3D geometry or 2D pixel features, respectively. Exp-c, d, e are results based on different feature fusion architectures for the geometry and pixel aligned features. Note that exp-b is the same as PIFu in Table 4.1, just named differently. Because essentially PIFu is a degenerate case of our method which only extracts pixel-aligned representations when learning implicit surface functions. Our results in the benchmarks are generated by the exp-e configuration. More analysis is in Section 4.5.2.

point encodings for occupancy estimation. To understand the advantages of our method, we separately show the impact of these two different types of features on clothed human mesh reconstruction. Results are reported in exp-a, b of Table 4.3. Ours-geometry and ours-pixel reconstruct human meshes using solely the 3D geometry or 2D pixel features, respectively. Exp-a shows significant improvement over exp-b in PSD, indicating that meshes reconstructed from 3D geometry features have better global topology robustness / regularities. This is attributed to global structure-aware 3D U-Net architectures and uniform coarse occupancy volume losses used to generate and supervise the latent voxel features. Visual comparisons in Figure 4.4 also verified this argument. However, meshes of exp-a are overly smooth lacking local surface details. This means that 2D pixel features focus more on learning high-frequency shape information such as clothes wrinkles. Moreover, by comparing exp-a, b with exp-e in

Table 4.3 and Figure 4.4, we observe that the best global and local performances are achieved only when 3D geometry and 2D pixel features are fused. Namely, the fused features contain the richest shape information *wrt.* each query point for implicit surface function-based dense, continuous occupancy fields learning.

Feature Fusion Architectures Knowing that the integrated representations lead to the best mesh reconstruction performance, we now further explore different 3D / 2D feature fusion architectures. Results are reported in exp-c, d, e of Table 4.3. Exp-c Ours-late-3D applies several fully connected (FC) layers upon the geometry-aligned 3D features before concatenating them with the pixel-aligned 2D features. Exp-d Ours-late-both applies separate FC layers on both the geometry and pixel aligned features before concatenating them. Exp-e Ours-early-3D directly concatenate the 3D / 2D features, as shown in Figure 7.2. Comparisons on CD, PSD, Cosine and $L2$ normal distances mean that the mesh reconstruction results are not quite sensitive to the fusion architectures. All these different feature fusion methods demonstrate clear improvement over exp-a, b which only use a single type of features for human mesh reconstruction. In our main benchmark results we use the configuration of exp-e for its balanced mesh and normal reconstruction performances and its simple implementation.

4.5.3 Failure Cases and Adversarial Training

Typical failure cases are shown in Figure 4.5. Plot-a is our method. Plot-b, c are meshes reconstructed from solely the 2D pixel features or the 3D geometry features, respectively. We have demonstrated in Figure 4.4 and Table 4.3 that mesh topology artifacts and wrong poses can be corrected leveraging global shape regularities from the geometry-aligned 3D features. However, in cases where 2D, 3D features (*i.e.* plot-b, c) both fail to generate correct meshes, the fused features (*i.e.* plot-a) will still lead to failures. To address this problem, we turn to 3D generative adversarial networks (3D-GAN). The implicit surface function training is based on sparse query points sampling and thus non-trivial to enforce 3D-GAN supervision directly on the whole continuous occupancy fields. Fortunately, the 3D U-Net based latent voxel

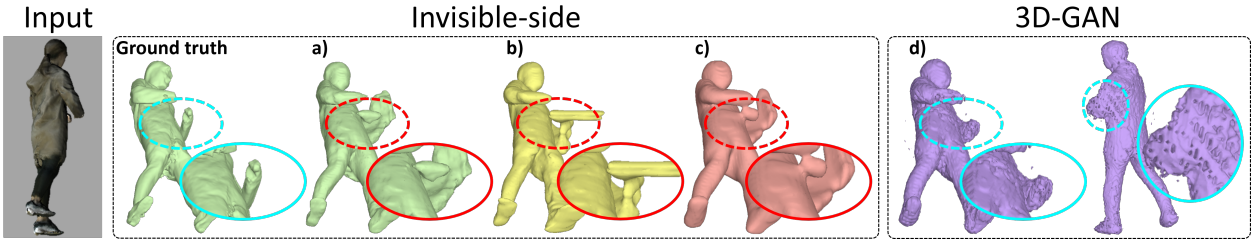


Figure 4.5: Typical failure cases. Plot-a is our method utilizing the early fusion based geometry and pixel aligned features. Plot-b, c are meshes reconstructed from solely the latent pixel or voxel features, respectively. Plot-d is similar to plot-c, but uses latent voxel representations trained jointly with coarse occupancy volume losses and 3D-GAN losses. When the latent pixel and voxel features, separately, both lead to human meshes with severe artifacts, the fused features struggle to correct all of these shape distortions. Further analysis on plot-d is presented in Section 4.5.3.

features are trained with coarse occupancy volume losses and naturally fit with voxel-based 3D-GAN losses. Human meshes reconstructed using only the 3D geometry features learned with both coarse occupancy volume losses and 3D-GAN losses are shown in plot-d of Figure 4.5. Its performances on the DeepHuman benchmark are: CD (2.464), PSD (3.372), Cosine (0.1298), $L2$ (0.4148). Comparing with plot-c, we do observe that "lumps" before the body's belly no longer exist but the mesh exhibits a new problem of having "honeycomb" structures. This phenomenon is also observed in the voxel-based 3D-GAN paper [WZX16]. Moreover, when we fuse such "honeycomb" 3D geometry features with the 2D pixel features to construct aligned query point features, the implicit surface function training fails to converge. We plan to have more studies on this issue as future work.

4.6 Discussion

We propose to interpolate and fuse geometry and pixel aligned query point features for deep implicit surface function-based single-view clothed human mesh reconstruction. Our method

of constructing query point encodings provides dedicated representation for each 3D point that captures its local shape patterns and resolves the feature ambiguity problem. Moreover, the latent voxel features, which we generate by structure-aware 3D U-Nets and supervise with coarse occupancy volume losses, help to regularize the clothed human mesh reconstructions with reduced shape and pose artifacts as well as distortions. The large-scale benchmark (10× larger then PIFu) and ablation studies demonstrate improved global and local performances of Geo-PIFu than the competing methods. We also discuss typical failure cases and provide some interesting preliminary results on 3D-GAN training to guide future work directions.

4.7 Additional Results

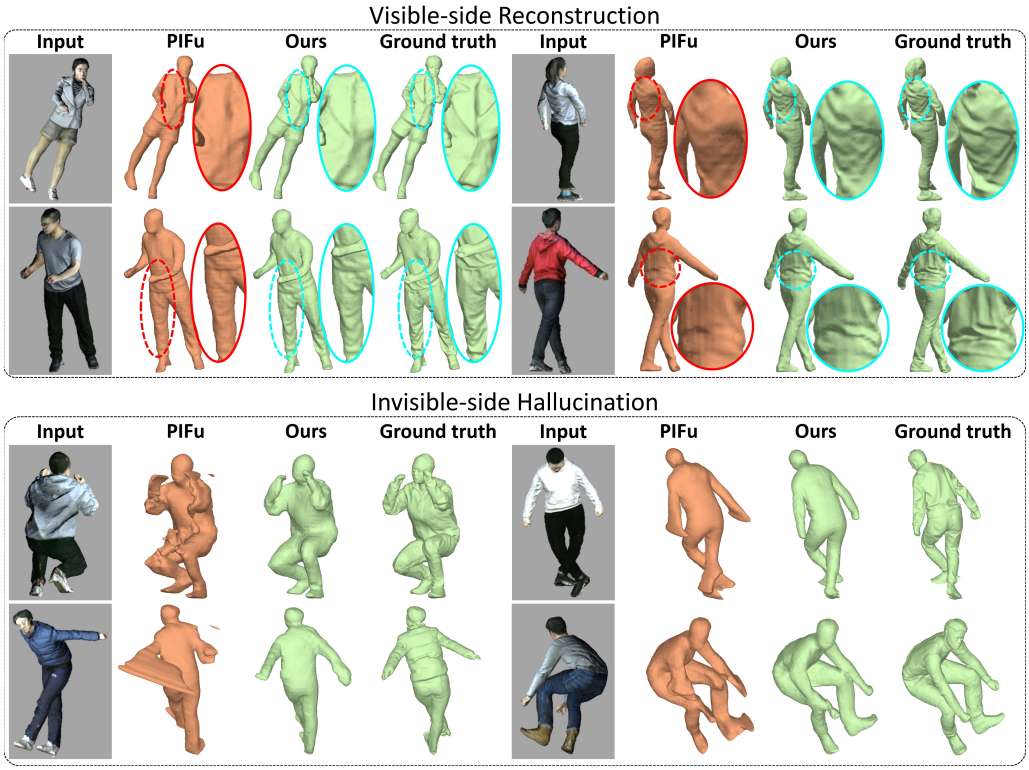


Figure 4.6: Single-view clothed human mesh reconstruction. Our results have less shape artifacts and distortions than PIFu. PIFu lacks global robustness when hallucinating invisible-side meshes of complex poses and large self-occlusion. Besides improved global regularities and better alignment with ground truth, our meshes also contain more accurate and clear local surface details than PIFu.

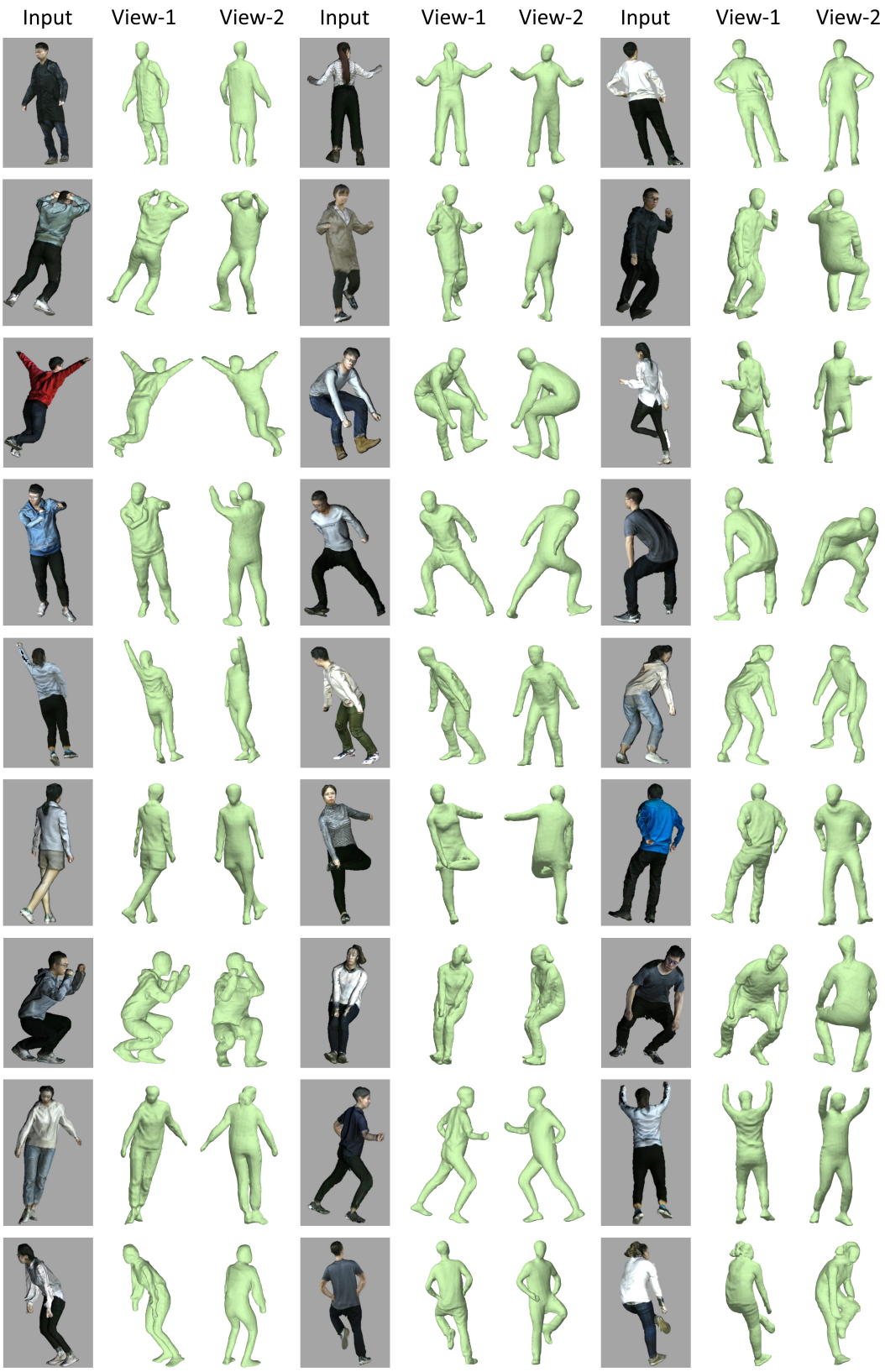


Figure 4.7: Single-view clothed human mesh reconstruction of our method Geo-PIFu.

CHAPTER 5

Image-based Animatable 3D Shape Reconstruction

5.1 Introduction

Digital humans have become an increasingly important building block for numerous AR/VR applications, such as video games, social telepresence [ORF16, LSS18] and virtual try-on. Towards truly immersive experiences, it is crucial for these avatars to obtain higher level of realism that goes beyond the uncanny valley [MMK12]. Building a photorealistic avatar involves many manual works by artists or expensive capture systems under controlled environments [CCS15, GLD19, PTX21], limiting access and increasing cost. Therefore, it is vital to revolutionize reconstruction techniques with minimal prerequisite (*e.g.*, a selfie) for future digital human applications.

Recent human models reconstructed from a single image combine category-specific data prior with image observations [ZBX20, JNV20, XWL21]. Among which, template-based approaches [KBJ18b, KPB19b, XZT19, APT19b, BTT19a] nevertheless suffer from lack of fidelity and difficulty supporting clothing variations; while non-parametric reconstruction methods [SHN19a, ZYW19b, SSS20, HCJ20b], *e.g.*, using implicit surface functions, do not provide intuitive ways to animate the reconstructed avatar despite impressive fidelity. In the recent work ARCH [HXL20], the authors propose reconstructing non-parametric human model using pixel-aligned implicit functions [SHN19a] in a canonical space, where all reconstructed avatars are transformed to a common pose. To do so, a parametric human body model is exploited to determine the transformations. By transferring skinning weights, which encode

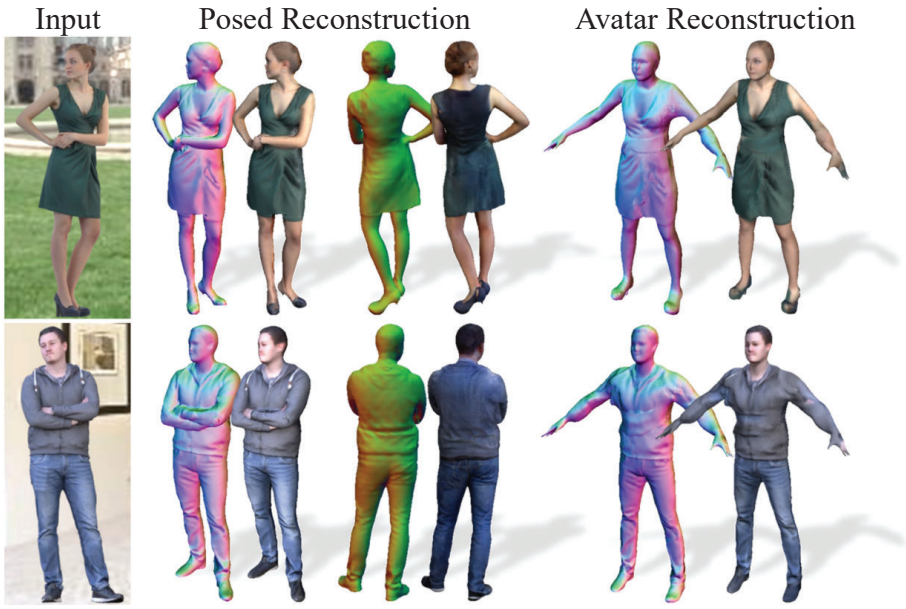


Figure 5.1: Given an image of a subject in arbitrary pose (left), our method could generate photorealistic avatars in both the posed input space (middle) as well as auto-rigged canonical space (right).

how much each vertex is influenced by the transformation of each body joint, from the underling body model, the reconstruction results are ready to animate. However, we observe that the advantages of a parametric body model and pixel-aligned implicit functions are not fully exploited.

In this paper we introduce ARCH++, which revisits the major steps of animatable avatar reconstruction from images and addresses the limitations in the formulation and representation of the prior work. First, current implicit function based methods mainly use hand-crafted features as the 3D space representation, which suffers from depth ambiguity and lacks human body semantic information. To address this, we propose an end-to-end geometry encoder based on PointNet++ [QSM17a, QYS17], which expressively describes the underlying 3D human body. Second, we find the unposing process to obtain the canonical space supervision causes topology change (*e.g.*, removing self-intersecting regions) and consequently the articulated reconstruction fails to obtain the same level of accuracy in the original posed space. Therefore, we present a co-supervising framework where occupancy is jointly predicted in both the

posed and canonical spaces, with additional constraints on the cross-space consistency. This way, we benefit from both: supervision in the posed space allows the prediction to retain all the details of the original scans; while canonical space reconstruction can ensure the completeness of a reconstructed avatar. Last, image-based avatar reconstruction often suffers from degraded geometry and texture in the occluded regions. To make the problem more tractable, we first infer surface normals and texture of the occluded regions in the image domain using image translation networks, and then refine the reconstructed surface with a moulding-inpainting scheme.

In the experiments, we evaluate ARCH++ on photorealistically rendered synthetic images as well as in-the-wild images, outperforming prior works based on implicit functions and other design choices on public benchmarks.

The contributions of ARCH++ include: 1) a point-based geometry encoder for implicit functions to directly extract human shape and pose priors, which is efficient and free from quantization errors; 2) we are the first to point out and study the fundamental issue of determining target occupancy space: posed-space fidelity vs. canonical-space completeness. Albeit ignored before, we outline the pros and cons of different spaces, and propose a co-supervising framework of occupancy fields in joint spaces; 3) we discover image-based surface attribute estimation could address the open problem of view-inconsistent reconstruction quality. Our moulding-inpainting surface refinement strategy generates 360° photorealistic 3D avatars. 4) our method demonstrates enhanced performance on the brand new task of image-based animatable avatar reconstruction.

5.2 Related Work

Template-based reconstruction utilizes parametric human body models, *e.g.*, SCAPE [ASK05] and SMPL [LMR15] to provide strong prior on body shape and pose to address ill-posed problems including body estimation under clothing [YFH16, ZPB17] and image-based human shape

reconstruction [BKL16, LRK17, KBJ18b, GK19, KPD19b, XJS19, KPB19b, XZT19]. While these works primarily focus on underling body shapes without clothing, the template-based representations are later extended to modeling clothed humans with displacements from the minimal body [PPH17], or external clothing templates [BTT19b], from 3D scans [YFH18, PPH17], videos [AMX18, HXZ20], and a single image [AMB19, BTT19b, JZH20]. As these approaches build clothing shapes on a body template mesh, the reconstructed models can be easily driven by pose parameters of the parametric body model. To address the lack of details with limited mesh resolutions, recent works propose to utilize 2D UV maps [LCT18, APT19b]. However, as a clothing topology can significantly deviate from the underling body mesh and its variation is immense, these template-based solutions fail to capture clothing variations in the real world.

Non-parametric capture is widely used to capture highly detailed 3D shapes with an arbitrary topology from multi-view systems under controlled environments [MBR00, BSB07, VBM08, TNM08, GSA09, VPB09, FP10, WVT12, TNM09, MNT12]. Recent advances of deep learning further push the envelope by supporting sparse view inputs [GVC18, HLC18], and even monocular input [LXS20]. For single-view clothed human reconstruction, direct regression methods demonstrate promising results, supporting various clothing types with a wide range of shape representations including voxels [VCR18b, JMT18b], two-way depth maps [GFM19, SLH19], visual hull [NSH19], and implicit functions [SHN19a, SSS20, HCJ20b]. In particular, pixel-aligned implicit functions (PIFu) [SHN19a] and its follow-up works [SSS20, HCJ20b] demonstrate impressive reconstruction results by leveraging neural implicit functions [MON19a, CZ19b, PFS19a] and fully convolutional image features. Unfortunately, despite its high-fidelity results, non-parametric reconstructions are not animation-ready due to missing body part separation and articulation. Recently, IF-Net [CAP20] exploits partial point cloud inputs and learns implicit functions using latent voxel features. Compared with image-based avatar reconstruction, completion from points can leverage directly provided strong shape and pose cues, and thus skip learning them from complex images.

Hybrid approaches combine template-based and non-parametric methods and allow us to leverage the best of both worlds, namely structural prior and support of arbitrary topology. Recent work [BST20] shows that using SMPL model as guidance significantly improves robustness of non-rigid fusion from RGB-D inputs. For single-view human reconstruction, Zheng et al. first introduce a hybrid approach of a template-model (SMPL) and a non-parametric shape representation (voxel [ZYW19b] and implicit surface [ZYL21]). These approaches, however, choose an input view space for shape modeling with reconstructed body parts potentially glued together, making the reconstruction difficult to animate as in the aforementioned non-parametric methods. The most relevant work to ours is ARCH [HXL20], where the reconstructed clothed humans are ready for animation as pixel-aligned implicit functions are modeled in an unposed canonical space. However, such framework fundamentally leads to sub-optimal reconstruction quality. We achieve significant improvement on accuracy and photorealism by addressing the hand-crafted spatial encoding for implicit functions, the lack of supervision in the original posed space, and the limited fidelity of occluded regions.

5.3 Methodology

Our proposed framework, ARCH++, uses a coarse-to-fine scheme, *e.g.*, initial reconstruction by learning joint-space implicit surface functions (see Fig. 5.2), and then mesh refinement in both spaces (see Fig. 5.3).

5.3.1 Joint-Space Implicit Surface Reconstruction

Semantic-Aware Geometry Encoder. The spatial feature representation of a query point is critical for deep implicit function. While the pixel-aligned appearance feature via Stack Hourglass Network [NYD16] has already demonstrated its effectiveness in detailed clothed human reconstruction by prior works [SHN19a, SSS20, HXL20, HCJ20b], an effective design of point-wise spatial encoding has not yet been well studied. The extracted geometry features

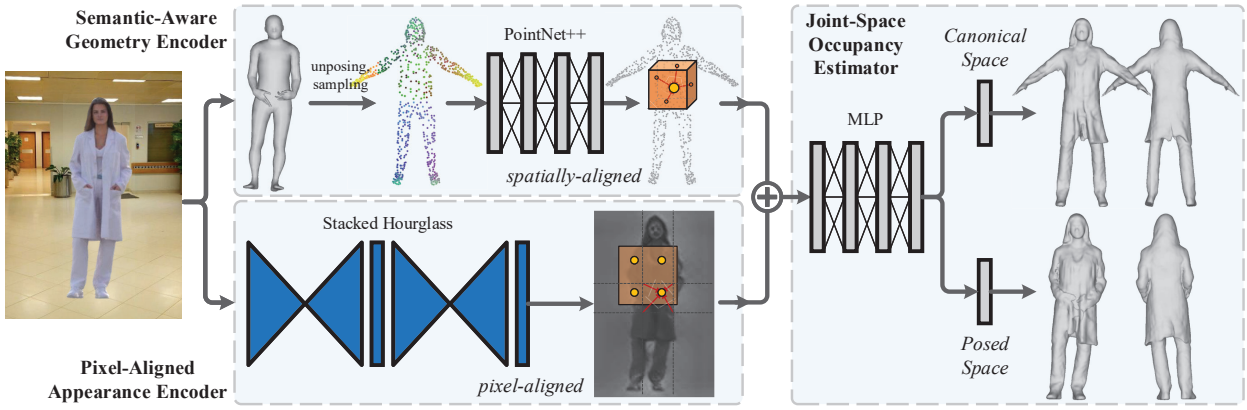


Figure 5.2: *Overview of the initial joint-space implicit surface reconstruction.* This procedure includes three components: i) semantic-aware geometry encoder, ii) pixel-aligned appearance encoder and iii) joint-space occupancy estimator. See text for detailed explanation.

should be informed of the semantics of the underlying 3D human body, which provide strong priors to regularize the overall dressed people shape.

The spatial encoding methods used previously include hand-crafted features (*e.g.*, RBF [HXL20]) and latent voxel features [CAP20, HCJ20b, ZYL21]. The former is constructed based on Euclidean distances between a query point and the body joints, ignoring the shapes. The voxel-based features capture both shape and pose priors of a parametric body mesh. Compared with the hand-crafted features, the end-to-end learned voxel features are better informed of the underlying body structures but often constrained by GPU memory sizes and suffer from quantization errors due to low spatial resolution. To effectively encode the shape and pose priors without losing any precision, we propose a novel semantic-aware geometry encoder that extracts point-wise spatial encodings. Essentially a parametric body mesh can be sampled into a point cloud and fed into PointNet++ [QSM17a, QYS17] to learn point-based spatial features, which have several advantages over both hand-crafted RBF features and voxel-based ones. Our method encodes both shape and pose priors from parametric shapes without computation overhead and quantization errors caused by the mesh voxelization process. Additional detailed statistical comparisons on points v.s. voxels in representing 3D shapes

are reported in [GFM19].

Given a parametric body mesh estimated and deformed by [XZT19, HXL20], we use a PointNet++ [QSM17a, QYS17] based semantic-aware geometry encoder to learn the underlying 3D human body prior. We sample N_0 (*e.g.*, 7324) points from the body mesh surfaces and feed them into the geometry encoder for spatial feature learning, that is,

$$f_{pn} : \{x_0^i\}_{i=1}^{N_0} \mapsto \{x_1^j, h_1^j\}_{j=1}^{N_1}, \{x_2^k, h_2^k\}_{k=1}^{N_2}, \{x_3^l, h_3^l\}_{l=1}^{N_3}, \tag{5.1}$$

where $x_0^i \in \mathbb{R}^3$ is a point sampled from the parametric body mesh. The PointNet++ based encoder utilizes fully-connected layers and neighborhood Max-Pooling to extract semantic-aware geometry features $h \in \mathbb{R}^{32}$ of a point. It also applies Furthest Point Sampling to progressively down sample the points $N_1 = 2048, N_2 = 512, N_3 = 128$ to extract latent features with increasing receptive fields. For example $\{x_1^j\}$ is a down sampled point set with size N_1, and $h_1^j \in \mathbb{R}^{32}$ is the learned feature w.r.t. each point.

As illustrated in Fig. 5.2, for any query point $p_a \in \mathbb{R}^3$ in the canonical space we obtain its point-wise spatial encoding $f_g \in \mathbb{R}^{96}$ via inverse $L2$-norm kernel based feature interpolation, followed by query coordinates concatenated Multi-layer Perceptrons (MLP). Particularly, we extract these features from different point set densities-j, k, l to construct concatenated features $f_g = (f_g^j \oplus f_g^k \oplus f_g^l)$ that are informed of multi-scale structures. For example, $f_g^j \in \mathbb{R}^{32}$ is defined as:

$$\begin{aligned} f_g^j(p_a, \{x_1^j, h_1^j\}) &= \text{MLP}(p_a \oplus \sum_m \frac{\|p_a - x_1^m\|^{-2}}{S(p_a, \{x_1^j, h_1^j\})} h_1^m), \\ S(p_a, \{x_1^j, h_1^j\}) &= \sum_m \|p_a - x_1^m\|^{-2}, \end{aligned} \tag{5.2}$$

where the index m is determined by finding the K nearest neighbors among the point set $\{x_1^j\}$ *w.r.t.* the query point. Empirically we found setting $K = 3$ obtains fair performance. The features extracted at other point set densities $f_g^k, f_g^l \in \mathbb{R}^{32}$ are obtained similarly leveraging $\{x_2^k, h_2^k\}$ and $\{x_3^l, h_3^l\}$, respectively.

Pixel-Aligned Appearance Encoder. We share the same architecture design as [SHN19a, SSS20, HXL20, HCJ20b] to map an input image $I \in \mathbb{R}^{512\times512\times3}$ into the latent feature maps

$\psi_\mu(I) \in \mathbb{R}^{128\times128\times256}$ via a Stacked Hourglass Network [NYD16] with weights μ. To obtain appearance encoding $f_a \in \mathbb{R}^{256}$ of any query point $p_b \in \mathbb{R}^3$ in the posed space, we project it back to the image plane based on a camera model of weak perspective projection, and bilinearly interpolate the latent image features:

$$f_a(p_b, I) = \mathcal{B}(\psi_\mu(I), \pi(p_b)), \tag{5.3}$$

where $\mathcal{B}(\cdot)$ indicates the differentiable bilinear sampling operation, and $\pi(\cdot)$ means weak perspective camera projection from the query point p_b to the image plane of I.

Joint-Space Occupancy Estimator. While most non-parametric and hybrid methods use the posed space as the learning and inference target space, ARCH instead reconstructs the clothed human mesh directly in a canonical space where humans are in a normalized A-shape pose. Different choices of the target space have pros and cons. The posed space is naturally aligned with the input pixel evidence and therefore the reconstructions have high data fidelity leveraging the direct image feature correspondences. Thus, many works choose to reconstruct a clothed human mesh in its original posed space (*e.g.*, PIFu(HD) [SHN19a, SSS20], Geo-PIFu [HCJ20b], PaMIR [ZYL21]). However, in many situations the human can demonstrate complex poses with self-intersection (*e.g.*, hands in the pocket, crossed arms) and cause a "glued" mesh that is difficult to articulate. Meanwhile, canonical pose reconstruction offers us a rigged mesh that is animation ready (via its registered A-shape parametric mesh [HXL20]). The problem of using the canonical space as the target space is that when we warp the mesh into its posed space there could be artifacts like intersecting surfaces and distorted body parts (see Fig. 5.6). Thus, the reconstruction fidelity of the warping obtained canonical-to-posed space mesh will degenerate. To maintain both input image fidelity and reconstruction surface completeness, we propose to learn the joint-space occupancy distributions.

We use a joint-space defined occupancy map O to implicitly represent the 3D clothed human under both its original posed space and a rigged canonical space:

$$O = \{(p_a, p_b, o_a, o_b) : \ p_a, p_b \in \mathbb{R}^3, \ -1 \leq o_a, o_b \leq 1\}, \tag{5.4}$$

where o_a, o_b denote the occupancy for points p_a and p_b. A point in the posed space is p_b and its mapped counterpart in the canonical space is $p_a = \text{SemDF}(p_b)$. The semantic deformation mapping (SemDF) between the original posed and the rigged canonical spaces is enabled by nearest neighbor-based skinning weights matching between p_b and the estimated underlying parametric body mesh [HXL20].

To enable mesh reconstruction in joint spaces, we use both point-wise spatial features $f_g \in \mathbb{R}^{96}$ that are informed of semantic full-body structures, and pixel-aligned features $f_a \in \mathbb{R}^{256}$ that encode human front-view appearances:

$$o_a = \mathcal{F}_\theta(f_g \oplus f_a),\ o_b = \mathcal{F}_\beta(f_g \oplus f_a), \tag{5.5}$$

where θ, β are network weights of the MLP-based deep implicit surface functions. To reconstruct avatars from the dense occupancy estimations in two spaces, we use Marching Cube [LC87a] to extract the isosurface at $o_a = \tau$ and $o_b = \tau$ (*e.g.*, $\tau = 0$), respectively.

The network outputs o_a, o_b are supervised by the ground truth joint-space occupancy $\hat{o}_a, \hat{o}_b$, depending on whether a posed space query point p_b and its corresponding canonical space point p_a are inside the clothed human meshes or not. Though p_a, p_b are a pair of mapped points their ground-truth occupancy values are not the same in all cases. For example, a point outside and close to the hand of a parametric body could has $\hat{o}_b > 0$ and $\hat{o}_a < 0$ if the original mesh in posed space has self-contact (*e.g.*, hands in the pocket). Namely, the SemDF defines a dense correspondence mapping between the two spaces but their occupancy values are not necessarily the same. Therefore, naively learning the distribution in one space and then warping the reconstruction into another pose can cause mesh artifacts (see Fig. 5.6). This motivates us to model two space occupancy distributions jointly in order to maintain both canonical space mesh completeness and posed space reconstruction fidelity.

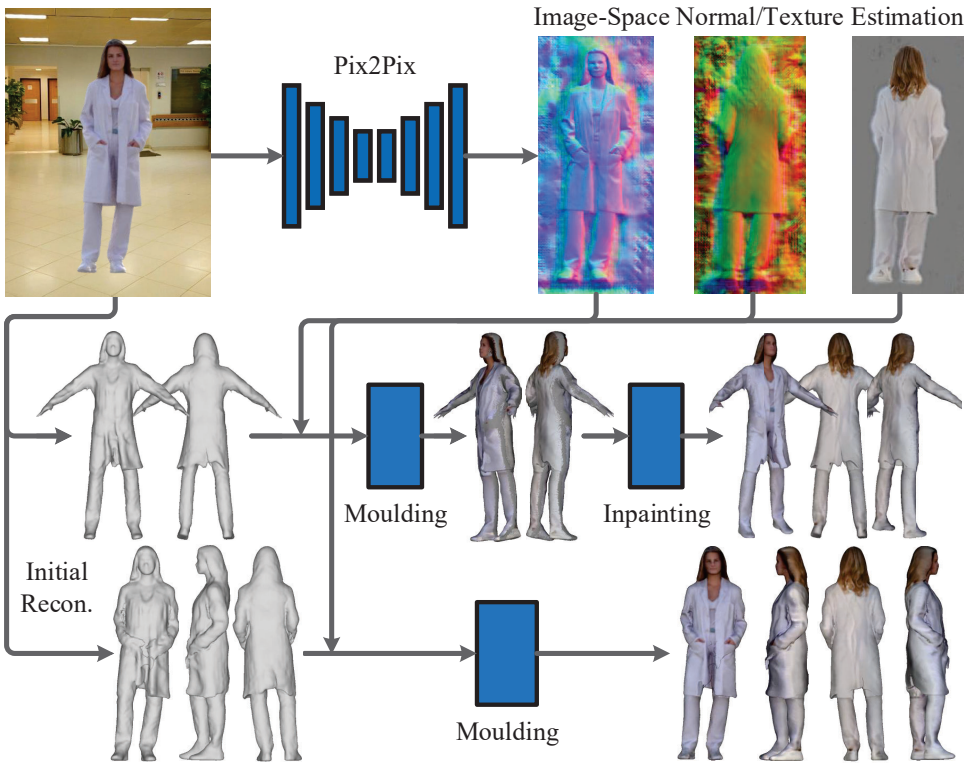


Figure 5.3: *Overview of the mesh refinement steps.* Our approach refines the initially estimated joint-space meshes from Fig. 5.2 using estimated normals and textures.

5.3.2 Mesh Refinement

We further refine the reconstructed meshes in joint spaces by adding geometric surface details and photorealistic textures. As illustrated in Fig. 5.3, we propose a moulding-inpainting scheme to utilize the front and back side normals and textures estimated in the image space. This is based on the observation that direct learning and inference of dense normal/color fields using deep implicit functions as [HXL20] usually leads to over-smooth blur patterns and block artifacts (see Fig. 5.5). In contrast, image space estimation of normal and texture maps produces sharp results with fine-scale details, and is robust to human pose and shape changes. These benefits are from well-designed 2D convolutional deep networks (*e.g.*, Pix2Pix [IZZ17, WLZ18]) and advanced (adversarial) image generation training schemes like GAN, with perceptual losses. The image-space estimated normal (and texture) maps could be used in two different ways. They can be used as either direct inputs into the Stack Hourglass as additional channels of the single-view image, or moulding-based front and back side mesh refinement sampling sources. In the experiments, we conduct ablation studies on these two schemes (*i.e.*, early direct input, late surface refinement) and demonstrate that our moulding-based refinement is better at maintaining fine-scale surface details across different views (see Fig. 5.8).

Posed Space. For the clothed human mesh obtained by Marching Cube in the original posed space, we conduct visibility tracing to determine if a vertex $V \in \mathbb{R}^3$ should be projected onto the front or the back side to bilinearly sample the normal/texture maps. Essentially, this is a moulding-based mesh refinement process for surface details and textures enhancement. We first conduct normal refinement. Note that for vertices whose unrefined normals $\mathbf{n} \in \mathbb{R}^3$ are near parallel (*i.e.*, within ε degrees) to the input image plane, we project them onto both the front and the back side normal maps $I_n^f, I_n^b \in \mathbb{R}^{512\times512\times3}$. We could then compute the refined surface normals $\mathbf{n}' \in \mathbb{R}^3$ via a linear blend fusion:

$$\begin{aligned} \mathbf{n}' &= \chi(1-\alpha')\,\mathcal{B}(I_n^f, \pi(V)) + \chi(\alpha')\,\mathcal{B}(I_n^b, \pi(V)), \\ \alpha' &= (90^\circ + \varepsilon - \alpha)/(2\varepsilon), \end{aligned} \tag{5.6}$$

where α is the angle between the unrefined normal and the forward camera raycast, and α' is the normalized value of α. Again, $\mathcal{B}(\cdot)$ indicates the bilinear sampling operation. The indicator function $\chi(\cdot)$ determines the blending weights of sampled normals from the front and the back sides:

$$\chi(\alpha') = \min(\max(\alpha', 0), 1) \tag{5.7}$$

This simple yet effective fusion scheme creates a normal-refined mesh with negligible blending boundary artifacts. With the refined surface normals we can further apply Poisson Surface Reconstruction [KH13b] to update the mesh topology but in practice we find this unnecessary since the moulding-refined avatar can already satisfy various AR/VR and novel-view rendering applications. This bump rendering idea is also used in DeepHuman [ZYW19b] but they only refine meshes using the front views. We further conduct the texture refinement in a similar manner but use the refined normals to help determine the linear blending weights of boundary vertices. Our moulding-based front/back normal and texture refinement method yields clothed human meshes that look photorealistic at different viewpoints with full-body surface details (*e.g.*, clothes wrinkles, hairs).

Canonical Space. The reconstructed canonical space avatar is rigged and thus can be warped back to its posed space and then refined via the same pipeline described above. However, a unique challenge for canonical avatar refinement is that mesh reconstructions in this space might contain unseen surfaces under the posed space. For example, in the third row of Fig. 5.5, the folded arm is in contact with the chest in the posed space but unfolded in the canonical space. Therefore, we do not have direct normal/texture correspondences of the chest regions of the canonical mesh. To address this problem, we render the front and the back side images of the canonical mesh with incomplete normal and texture, and treat it as an inpainting task. This problem has been well studied using deep neural networks [YTA20, YLY18] and patch matching based methods [BSC00, BSF09, HKA14]. We use PatchMatch [BSF09] for its robustness. As demonstrated in the last two columns of Fig. 5.5, compared to directly regressing point-wise normal and texture, our inpainting-based results obtain sharper details

and fewer artifacts.

5.3.3 Training Losses

The training process involves learning deep networks for two goals: joint-space occupancy estimation with $\mathcal{L}_o$, and normal/texture estimation with $\mathcal{L}_n$ and $\mathcal{L}_t$. Specifically, $\mathcal{L}_o$ is the occupancy regression loss of our joint-space deep implicit functions, and $\mathcal{L}_n, \mathcal{L}_t$ are image translation losses of the normal, texture estimation networks.

Joint-space Occupancy Estimation. The deep implicit function training is based on query point sampling and supervised occupancy regression with Tanh output layers. We randomly sample mesh points p_a, p_b in two spaces and then add diagonal Gaussian perturbation with a standard deviation of 5 cm to increase the sample coverage of close-to-surface regions in space. In each training iteration we sample 20480 pairs of query points (p_a, p_b), with predicted occupancy (o_a, o_b). The joint-space occupancy regression loss contains three terms:

$$\mathcal{L}_o(o_a, o_b) = \mathcal{L}_o^{occ}(o_a) + \mathcal{L}_o^{occ}(o_b) + \mathcal{L}_o^{con}(o_a, o_b), \tag{5.8}$$

where $\mathcal{L}_o^{occ}(o_a), \mathcal{L}_o^{occ}(o_b)$ denote the Smooth $L1$-Loss between the estimated occupancy values and their ground truth in the canonical and the posed spaces, respectively. $\mathcal{L}_o^{con}(o_a, o_b)$ is a contrastive loss regularizing the occupancy consistency between the two spaces, that is,

$$\mathcal{L}_o^{con}(o_a, o_b) = \begin{cases} |o_a - o_b|, & \text{if } \hat{o}_a = \hat{o}_b, \\ \lambda_1 \max(\lambda_2 - |o_a - o_b|, 0), & \text{otherwise}, \end{cases} \tag{5.9}$$

where λ_1 and λ_2 are two parameters to adjust the penalty of inconsistent joint-space groundtruth pairs. Those pairs usually exist around the self-intersecting regions and need to be down-weighted due to the errors in canonical space supervision. Empirically, we set $\lambda_1 = 0.1$ and $\lambda_2 = 0.3$.

Mesh Refinement. We consider the image-space normal and texture estimation as an image-to-image translation task. Given an input image I, our task is to learn the front

Components	Posed Space			Canonical Space			Mean		
	Normal ↓	P2S ↓	Chamfer ↓	Normal ↓	P2S ↓	Chamfer ↓	Normal ↓	P2S ↓	Chamfer ↓
Posed Sup. Only	0.037	0.674	0.787	0.087	1.898	1.597	0.062	1.286	1.192
Canonical Sup. Only	0.039	0.716	0.838	0.046	0.606	0.997	0.043	0.661	0.917
Joint	0.037	0.662	0.789	0.045	0.620	0.988	0.041	0.641	0.825
Joint + GeoEnc	0.033	**0.495**	**0.614**	0.040	**0.471**	**0.819**	0.036	**0.483**	**0.717**
Joint + GeoEnc + Refine	**0.031**	**0.495**	**0.614**	**0.039**	**0.471**	**0.819**	**0.035**	**0.483**	**0.717**

Table 5.1: *Ablation studies on the effectiveness of ARCH++ proposed components in both spaces: posed vs. canonical.* Best scores are in **bold**. Rows are target reconstruction spaces, columns are evaluation spaces. The first row means using the posed space as the target space (*e.g.*, PIFu, PIFuHD, Geo-PIFu, PaMIR), whose reconstruction can be warped into the canonical space via a registered parametric body to compute evaluation metrics in both spaces. The second row means direct supervision and reconstruction in the canonical space, followed by warping into the posed space (*e.g.*, ARCH). The rest rows are based on our joint-space co-supervision and reconstruction scheme.

normal map I_n^f, the back normal map I_n^b and the back side texture map $I_t^b \in \mathbb{R}^{512\times512\times3}$. Note we assume the input image can be directly used as the front texture map. Inspired by the demonstrated superior results of Pix2Pix [IZZ17, WLZ18], we define the training losses as:

$$\begin{aligned} \mathcal{L}_n(I_n^f, I_n^b) &= \mathcal{L}_n^{rec}(I_n^f) + \mathcal{L}_n^{rec}(I_n^b) + \mathcal{L}_n^{vgg}(I_n^f) + \mathcal{L}_n^{vgg}(I_n^b), \\ \mathcal{L}_t(I_t^b) &= \mathcal{L}_t^{rec}(I_t^b) + \mathcal{L}_t^{vgg}(I_t^b) + \mathcal{L}_t^{adv}(I_t^b), \end{aligned} \tag{5.10}$$

where $\mathcal{L}^{rec}(\cdot)$ denotes the $L1$ distance reconstruction loss, $\mathcal{L}^{adv}(\cdot)$ means the generative adversarial loss and $\mathcal{L}^{vgg}(\cdot)$ is the VGG-perceptual loss proposed by [JAF16]. In the experiments, we found that the generative adversarial loss $\mathcal{L}^{adv}(\cdot)$ counteracts to performance in the normal map estimation task and thus we only enforce this loss term upon the back side texture map. One explanation is that the normal map space is more constrained and has fewer variations than the texture map, and therefore adversarial training does not fully show its effectiveness in this case.

5.4 Implementation Details

We implement our framework using PyTorch and conduct the training with one NVIDIA Tesla V100 GPU. The proposed deep neural networks are trained with RMSprop optimizer with a learning rate starting from 1e-4. We use an exponential learning rate scheduler to update it every 3 epochs by multiplying with the factor 0.1 and terminate the training after 12 epochs.

5.5 Experiments

In this section, we present the experimental settings, result comparisons and ablation studies of ARCH++.

5.5.1 Datasets

We adopt the dataset setting from [HXL20, SSS20]. Our training dataset consists of 375 3D scans from RenderPeople dataset [REN] and 205 3D scans from AXYZ dataset [SR]. These watertight human meshes have various clothes styles as well as body shapes and poses. Our testing set includes 37 scans from RenderPeople dataset [REN], 192 scans from AXYZ dataset, 26 scans from BUFF dataset [ZPB17], and 2D images from Internet public domains, representing clothed people with a large variety of complex clothes. The subjects in the training dataset are mostly in standing pose, while the subjects in the test dataset contain various poses including sitting, twisted and standing, as well as self-glued and separated limbs. We use Blender and 38 environment maps to render each scan under different natural lighting conditions. For each 3D scan, we generate 360 images by rotating a camera around the mesh with a step size of 1 degree. These RenderPeople images are used to train both the occupancy estimation and the image translation networks.

We generate ground truth clothed human meshes in the canonical pose using the method

Methods	RenderPeople			BUFF		
	Normal ↓	P2S ↓	Chamfer ↓	Normal ↓	P2S ↓	Chamfer ↓
BodyNet [VCR18b]	0.26	5.72	5.64	0.31	4.94	4.52
VRN [JMT18b]	0.12	1.42	1.60	0.13	2.33	2.48
SiCloPe [NSH19]	0.22	3.81	4.02	0.22	4.06	3.99
IM-GAN [CZ19b]	0.26	2.87	3.14	0.34	5.11	5.32
PIFu [SHN19a]	0.11	1.45	1.47	0.13	1.68	1.76
PIFuHD [SSS20]	0.11	1.37	1.43	0.13	1.63	1.75
ARCH [HXL20]	0.04	0.74	0.85	0.04	0.82	0.87
ARCH++ [Ours]	**0.03**	**0.50**	**0.61**	**0.03**	**0.58**	**0.61**

Table 5.2: *Quantitative results and comparisons of normal, P2S and Chamfer errors between posed reconstruction and ground truth on RenderPeople and BUFF datasets.* Best scores are in **bold**.

introduced in [HXL20]. Note that the warping process between the posed and the canonical spaces inevitably contain model noises (*e.g.*, self-contact region artifacts, skinning weights nearest neighbor discontinuities), which motivates our joint-space co-supervision and reconstruction scheme.

5.5.2 Results and Comparisons

We use the same metrics as [SHN19a, SSS20, HXL20] for quantitative evaluation of the reconstructed meshes. We report the average point-to-surface Euclidean distance (P2S) and the Chamfer distance in centimeters, as well as the $L2$ normal re-projection errors. The two state of the art methods for our main comparisons are PIFuHD [SSS20] and ARCH [HXL20], both are built upon PIFu [SHN19a] with improvements in different aspects. PIFuHD ingests

[1]Our results (green boxes) have fewer reconstruction artifacts (*e.g.*, incorrect normal directions, mesh distortion) than ARCH (red boxes).

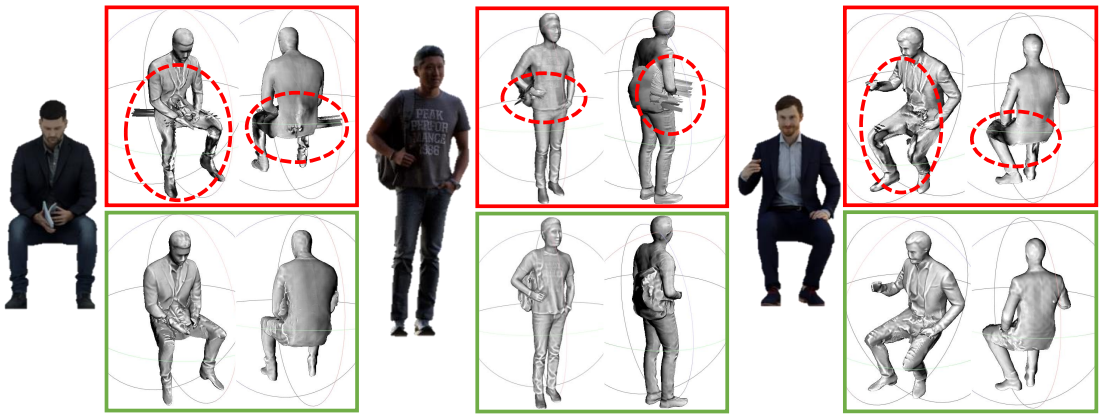


Figure 5.4: We intentionally hide the method names for you to have a fair comparison on your own (*please zoom in*). The answers are[1].

high-resolution images in a sliding window manner to achieve rich surface reconstruction details. ARCH leverages nearest neighbor-based linear blend skinning weights and hand-crafted RBF features to reconstruct animatable avatars in a canonical space. In addition to these two most related methods, we also include multiple prior methods [VCR18b, JMT18b, NSH19, CZ19b, SHN19a] and report the benchmark results on the RenderPeople and the BUFF datasets in Tab. 5.2. ARCH++ [Ours] results outperform the second best method ARCH by large gaps.

The visual comparisons in Fig. 5.5 and Fig. 5.4 further explain the advantages of our improvement. PIFuHD suffers from shape distortions due to lacking shape and pose priors provided by the end-to-end geometry encoder. Note that PIFuHD is incapable of reconstruct canonical space avatars and lacks texture estimation. ARCH reconstructions tend to be over smooth and blurry. Its recovered mesh normal and texture also have several block artifacts. Additionally, both methods fail to hallucinate plausible back-side surface details like clothes wrinkles, hairs, *etc.* In comparison, our approach achieves photorealistic and animatable reconstructions in joint spaces and across different viewpoints. We further show our results on Internet images in Fig. 5.9.

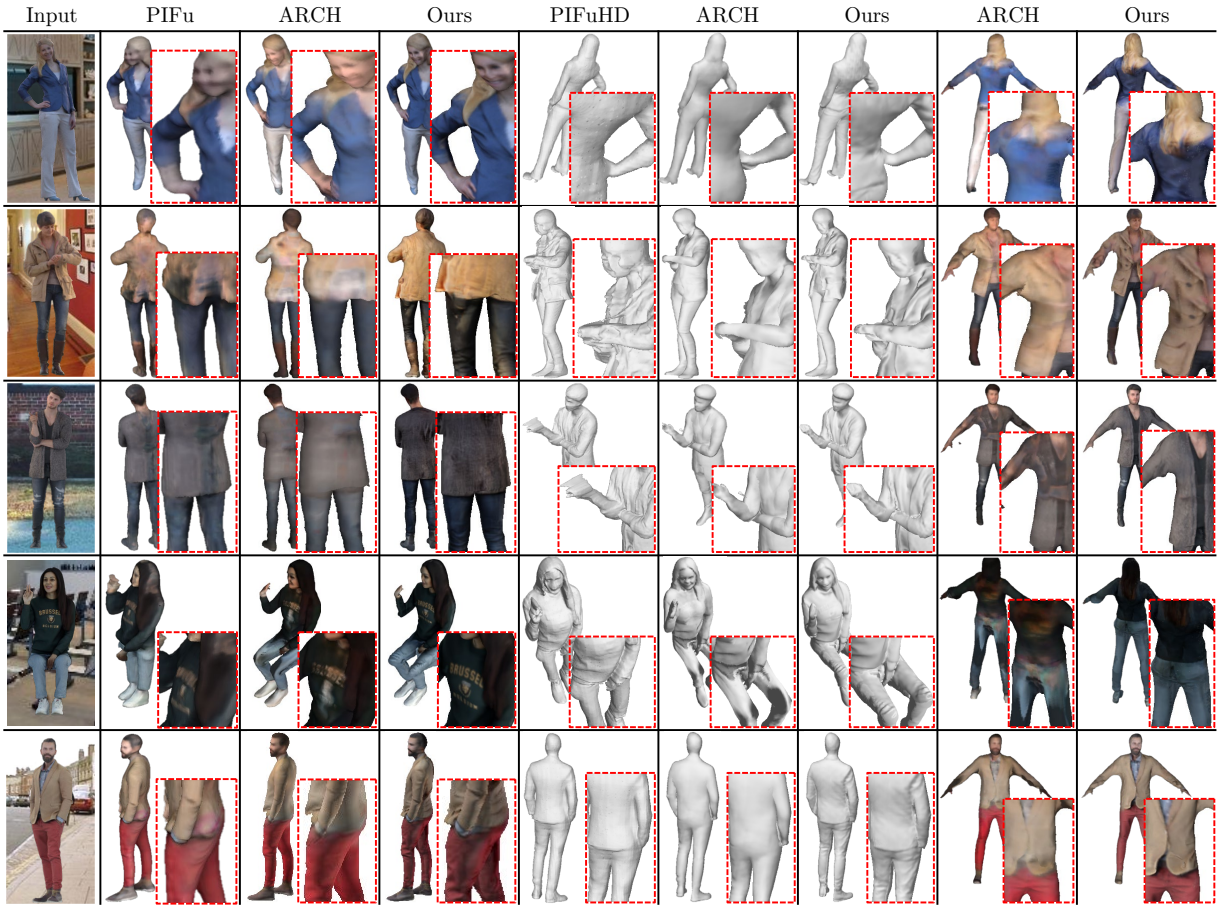


Figure 5.5: *Qualitative comparisons against the state-of-the-art methods [SHN19a, SSS20, HXL20].* The first column is input. Column 2-4, 5-7 are color and shape reconstruction results, respectively, in the posed space. The last two columns are canonical space avatar reconstructions. Our method handles arbitrary poses with self-contact and occlusions robustly, and reconstructs a higher level of details than existing methods.

5.5.3 Ablation Studies

Joint Space Reconstruction. To further understand the impact of the proposed methods, we present ablation studies in Tab. 5.1. The first three rows demonstrate the effectiveness of joint-space co-supervision, achieving balanced performances on both the posed and the canonical space mesh reconstructions. Choosing the posed space as the reconstruction target space (*e.g.*, PIFu, PIFuHD, Geo-PIFu, PaMIR) can cause missing surfaces and topology distortions in the posed-to-canonical space warped meshes (see Fig. 5.6). Meanwhile, choosing

Variants	Normal ↓	P2S ↓	Chamfer ↓
Depth [SHN19a]	0.047	0.78	0.93
RBF [HXL20]	0.042	0.74	0.85
End-to-end Voxel [HCJ20b, ZYL21]	0.034	0.52	0.63
End-to-end Point	**0.033**	**0.50**	**0.61**

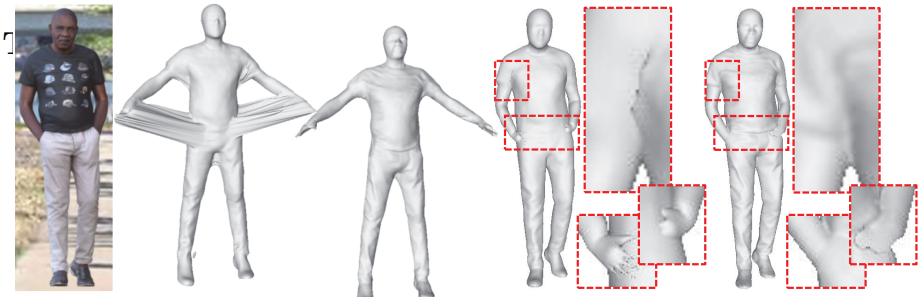


Figure 5.6: *Ablation studies on the reconstruction space.* Single-space reconstruction shows artifacts of either mesh surface over-stretching or intersecting surfaces when warping from one space to another. Our joint-space reconstruction obtains balanced performance of both high reconstruction completeness under the canonical space and high input image fidelity under the posed space.

the canonical space as the target space (*e.g.*, ARCH) can cause self-intersecting meshes with broken manifold as well as body part un-natural deformations in the canonical-to-posed space warped meshes. In contrast, our co-supervision and joint-space inference methods achieve both reconstruction fidelity in the posed space and body mesh completeness in the canonical space.

Geometry Encoding. As shown in Tab. 5.1, we observe further error reduction leveraging the end-to-end learned point-wise spatial encodings. The prior method ARCH uses handcrafted RBF features that only model the pose prior of parametric body mesh skeletons, ignoring the mesh shape. In comparison, our point-based features are informed of both pose and shape priors of the underlying parametric body model w.r.t. a clothed human

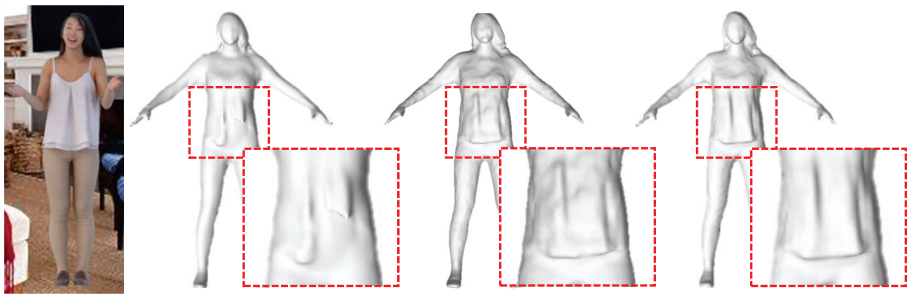


Figure 5.7: *Ablation studies on geometry encoding.* Learned spatial features capture both pose and shape priors of the underlying parametric models and thus enable mesh reconstruction with more surface details than the handcrafted RBF features. Meanwhile, results of the voxel-based features are noisier than the point-based ones due to mesh quantization (*i.e.*, voxelization) errors.

Figure 5.8: *Ablation studies on normal refinement*: Object-space Regression (OS Reg.), Image-space Input (IS Input) and Image-space Regression (IS Reg.). Our method IS Reg. leads to rich reconstruction details (*e.g.*, clothes wrinkles) in all views.

Figure 5.9: *An application of digital human capture from photos.*

Variants	Posed ↓	Canonical ↓	Mean ↓
Baseline	0.033	0.040	0.037
Object-space Regression [HXL20]	0.032	0.041	0.037
Image-space Input [SSS20]	0.032	**0.038**	**0.035**
Image-space Regression	**0.031**	0.039	**0.035**

Table 5.4: *Ablation studies on different ways of normal refinement.*

mesh, and thus improve the surface reconstruction quality. We further implement the learned volumetric spatial feature encodings used in Geo-PIFu and PaMIR as an alternative encoder and inject into our framework for direct comparisons. The results are shown in Tab. 5.3 and Fig. 5.7. While both types of end-to-end spatial features outperform the hand crafted RBF features, our point-based feature extraction method does not suffer from computation overhead and mesh quantization errors of the voxel-based approach.

Normal Refinement. While single-image based direct inference of human meshes with rich surface details at both the front and the back side remains an open question, some empirical observations and prior works indicate that normal estimation is a relatively easier task and can help refine the reconstructions. In Tab. 5.4 and Fig. 5.8 we experiment on three principle ways of leveraging the estimated normals for mesh reconstructions with refined surface details. Among these normal refinement methods, our front/back-side image space normal regression and moulding-based surface refinement approach outperforms other variants. Object-space normal regression is adopted in ARCH and is based on learning deep implicit functions of spatial normal fields. It fails to generate rich back side details and sometimes causes block artifacts as shown in the fourth row of Fig. 5.5. Image-space input is used in PIFuHD. It concatenates the color image input with estimated image-space normal maps and feeds them into Stack Hourglass for feature extraction. While this method achieves the same level of quantitative performance as our mesh refinement approach, its visual results are not as sharp as ours at both the front and the back sides. A degenerated case of our mesh refinement method is studied before in DeepHuman where they only estimate front-view normal maps and therefore lack reconstruction details at the back side.

5.6 Discussion

In this paper, we revisit the major components in existing deep implicit function based 3D avatar reconstruction. Our method ARCH++ produces results which have high-level fidelity

and are animation-ready for many AR/VR applications. We conduct a series of comparisons with and analysis on the state of the art to validate our findings. For future works, we plan to incorporate environment information (e.g., lighting, affordance) to further understand the body pose and appearance, and address current limitations.

CHAPTER 6

Learning Self-Supervised Deep Voxel Representation

6.1 Introduction

A physical scene is far more complex than any number of images of it, so it is challenging to just "reconstruct *it*". The question of how to evaluate a *model* of a scene depends on whether one has access to additional sensors (*e.g.* tactile), or prior knowledge (*e.g.* scale of objects). In the absence of any side information, one often utilized measure of quality of a model built from data is its ability to predict data that the model can generate [AMH78, Ast79]. Hence, view synthesis is a critical step in building and evaluating models of physical scenes from images. There are also practical ramifications of novel-view synthesis to video compression, graphics rendering and reinforcement learning. Before deep learning, novel-view synthesis was either approached as a pipeline of motion estimation, sparse reconstruction, topology estimation, meshing, and texture mapping [KLT06], or directly by resampling the plenoptic function [GGS96]. More recently, Sitzmann *et al.* [STH19a] proposed DeepVoxels – an approach employing a 3D grid of persistent features integrated over input images along with 2D lifting and projection networks.

We learn 3D voxel embeddings of object shape and appearance based on image patches, whose pose can be directly controlled to generate novel views at high resolution. Our method is based on DeepVoxels [STH19a], but with improved rendering quality leveraging a series of enhancements and a simple yet effective implementation trick of 3D embeddings sampling. Specifically, DeepVoxels++ is different from DeepVoxels in four aspects:

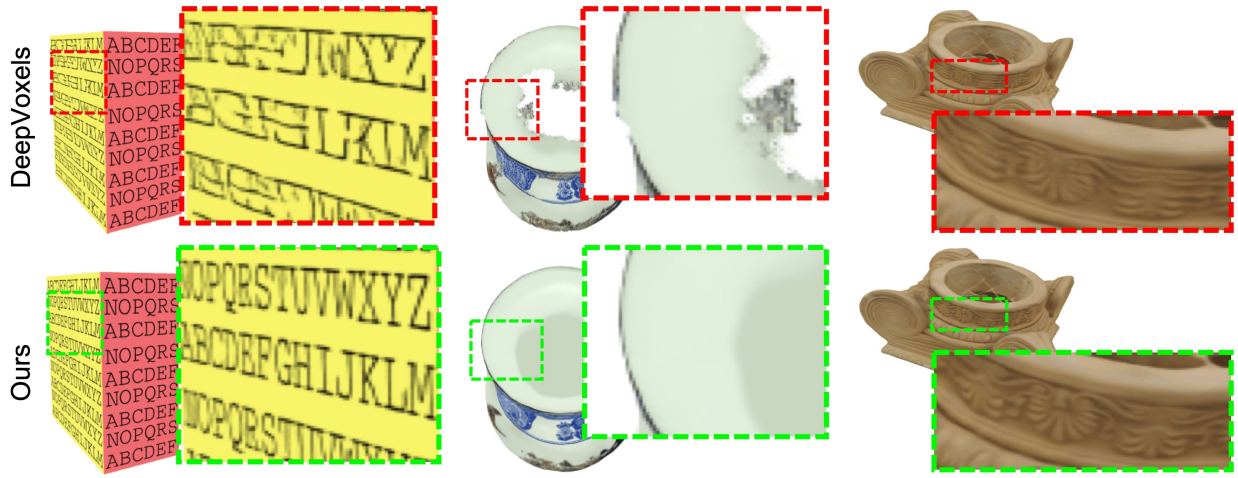


Figure 6.1: Rendering results of our model have sharper details (*e.g.* text, fine-grained shapes) and fewer artifacts (*e.g.* aliasing, holes) than DeepVoxels [STH19a].

1. Low-complexity patch modelling. We adopt a patch-based training and inference scheme that halves the 2D U-Net parameters used in image feature extraction and neural rendering. It reduces the complexity of large image context modeling (*e.g.* 512×512×3 full image vs. 128×128×3 small patch) by learning local patterns, and enables image modeling as well as rendering at high resolution in sliding window manner.

2. View-dependent voxel feature transformations. Viewpoint changes can cause perspective transformations in the images (see Fig. 6.3). We directly learn view-dependent feature transformation kernels in the lifting/projection steps to model such perspective effect. We transform the features from input patches to the 3D voxel embeddings and then from the voxels to output patches based on the relative voxel-camera poses. We demonstrate this idea on objects of diffuse reflection, delicate shapes and limited training views.

3. Recurrent-concurrent voxel feature aggregation. We aggregate 3D voxel embeddings utilizing both recurrent gated-fusion and concurrent max-pooling. It differs from existing works which treat multi-view images as a sequence and solely rely on recurrent networks [TCF19, STH19a, CVG14]. Our method increases surface coverage of an object during each iteration of voxel feature aggregation and improves data utilization rate. For example, our

model can outperform DeepVoxels [STH19a] under different training data size configurations.

4. Frustum representation sufficient sampling. Sampling the 3D voxel embeddings into a frustum of the target pose is a critical step in decoding the learned volumetric representation into a rendered image. We empirically found that sufficient frustum sampling is a simple yet effective implementation trick to alleviate the issue of limited voxel resolution, reduce blurring artifacts, and preserve sharp details. It enforces the voxel feature learning process which in turn helps encode fine-scale details in the learned 3D voxel embeddings.

Overall, our approach improves over DeepVoxels [STH19a] upon the visual quality of novel-view rendering at various poses (by up to 33% SSIM error reduction and 22% PSNR performance boost). We use the same 360° novel-view synthesis benchmarks as [STH19a], which contain 512×512 color images of delicate shapes and textures. In contrast, other object based novel-view synthesis methods [ZTS16, PYY17, SHL18, OTW19] mainly use the 256×256 ShapeNet images that consist of mostly mono-color flat surfaces and do not evaluate rendering results at 360° densely sampled poses.

6.2 Related Work

Our work is related to multiple fields in 3D computer vision and graphics: image-based modeling, deep learning for view generation, 3D representation learning and neural-rendering, *and* deep learning with feature structure constraints.

Image-based modeling Image-based modeling and rendering techniques [KLT06] are the early approaches to the novel view synthesis problem. Modern approaches, such as [HAS17, HPT18, PZ17], are able to obtain high-quality results even for challenging scenarios with hand-held cameras. However these methods usually require multiple steps to (soft) reconstruct the object or learn image blending weights, and therefore they are prone to accumulative errors. They do not take full advantage of large scale multi-view datasets for 3D latent embedding learning and (adversarial) image generation training from the learned

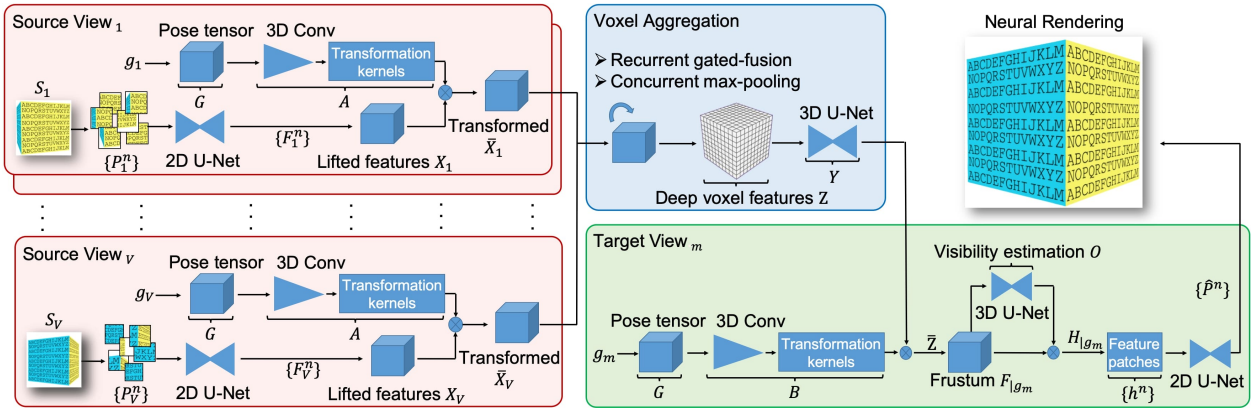


Figure 6.2: DeepVoxels++ pipeline. Red: view-dependent patch feature extraction from V views. Blue: 3D voxel embeddings aggregation with recurrent gated-fusion and concurrent max-pooling. Green: view-dependent image patch rendering. The networks are trained jointly with L^1 image reconstruction losses upon rendered views.

embeddings.

Deep learning for view generation With the advent of deep convolutional neural networks (CNNs), data-driven methods are gaining popularity for novel view generation [FNP16, JKM17, ZTS16, MGK19, YWW18, PYY17, SHL18, ZTF18, FBD19, STH19a, SZW19, TZ19, STB19, TZT18]. The early methods overlook inherent 3D object structures/constraints and rely heavily on optical flow estimation [ZTS16] and generative adversarial networks [YWW18]. The former can maintain fine details in generated images while the latter are good at handling large pose changes for view synthesis. There are also hybrid approaches that combine the benefits of both sides [PYY17, SHL18]. However, these methods usually lack a learned latent representation of the object that is geometrically persistent, and thus tend to produce inconsistent images across output poses [STH19a, STH19b].

3D representation learning and rendering 3D representation learning and neural-rendering with deep networks is a problem studied in 3D Deep Learning. Various approaches have been proposed using point clouds [QSM17b], implicit neural functions [SZW19], voxel grids [KHM17], multi-plane images [MSO19, ZTF18, FBD19, STB19, KWR16], and *etc.* We

follow the line of work using voxel grids [KHM17, TCF19, OTW19, STH19a] which offer a geometrically persistent structure to integrate visual information across multiple poses around the object. In particular Sitzmann *et al.* [STH19a] demonstrate promising results for novel-view rendering utilizing a learned deep voxel representation. In this work, we achieve greatly improved visual quality for 360° novel-view synthesis than [STH19a] via a series of enhancements on feature extraction, transformation and aggregation, *and* a simple yet effective implementation trick of voxel embeddings sufficient sampling when rendering images.

Learning with feature structure constraints Our work is also related to the emerging direction of introducing explicit structure constraints upon deep features to data-driven deep networks [YRY15, WGT17, NLT19, XCJ19]. For example, Worrall *et al.* [WGT17] impose a 3×3 rotation matrix constraint on high-dimensional features by length dividing and sub-vector multiplication to learn an interpretable representation for rotation/scaling factors. In this work, we propose to learn voxel feature transformation kernels conditioning on the relative voxel-camera poses. The learned kernels are used to model perspective transformations of the observed/rendered images induced by viewpoint changes under diffuse reflectance.

6.3 Methodology

Our model, DeepVoxels++, learns latent 3D voxel embeddings using color images of an object from multiple viewpoints. Our deep network architecture can be perceived as: an encoder-decoder with a geometrically consistent voxel feature space as the latent representation. As shown in Fig. 6.2, the architecture comprises three stages that are trained jointly by 2D view prediction without any 3D occupancy supervision: (*encoder*) view-dependent feature extraction from image patches, (*bottleneck*) recurrent-concurrent aggregation of lifted features to form the latent 3D voxel embeddings, (*decoder*) view-dependent patch rendering. At *test time* we only need the learned voxel embeddings (*bottleneck*) and the view-dependent patch neural-rendering network (*decoder*) for novel-view synthesis.

The training data of each object consists of M multi-view images $\{I_i, g_i\}_{i=1}^{M}$, where I_i is a 512×512×3 image captured at a pose g_i. At training time, the multi-view images are sampled into tuples of $\{S_i, T_i^0, T_i^1\}_{i=1}^{M}$. During each training step, the networks are updated with L^1 reconstruction losses upon the predicted target views $\{(\hat{T}_j^0, \hat{T}_j^1)\}_{j=1}^{V}$, accepting multiple source images $\{S_j\}_{j=1}^{V}$ as input. We aggregate information from V (*e.g.* 1, 4, 8, and *etc.*) views concurrently during training, making use of 3D-GRU and max-pooling at the bottleneck stage. This training methodology is to ensure large coverage of the object surface within each recurrent-concurrent step of 3D voxel embeddings aggregation. A degenerate case is DeepVoxels, which only conducts recurrent aggregation (strictly $V = 1$); *i.e.* without concurrent consideration of views. The previous training strategy induces single-view observation caused latent voxel feature update bias, and thus has low data utilization efficiency.

6.3.1 View-dependent patch feature extraction

To learn deep 3D voxel embeddings from multi-view images, we first sample image patches from training images in sliding window manner. We then extract 2D feature maps from the set of image patches and accumulate these features in voxel space of pre-defined resolution, via view-dependent feature lifting.

Patch feature extraction We subdivide each source image S_i into small-size patches $\{P_i^n\}_{n=1}^{N}$ via a sliding window with overlaps. Patches are encoded via a 2D U-Net with skip connections for feature extraction: $P_i^n \mapsto F_i^n$. For very large images, if GPU memory sizes prohibit training on all N patches at one pass, we can sample a subset $\{P_i^n\}_{n=1}^{N'}$. In our experiments, we randomly sample 80% patches, but note the possibility of sampling heuristically *e.g.* sampling patches containing high-frequency/fine-scale content more frequently [SQA15]. Compared with the full-image based prior methods, the patch-based scheme enables our method to better learn (and render) local image patterns.

View-dependent feature lifting We first define a s×s×s cubic voxel space for aggregating

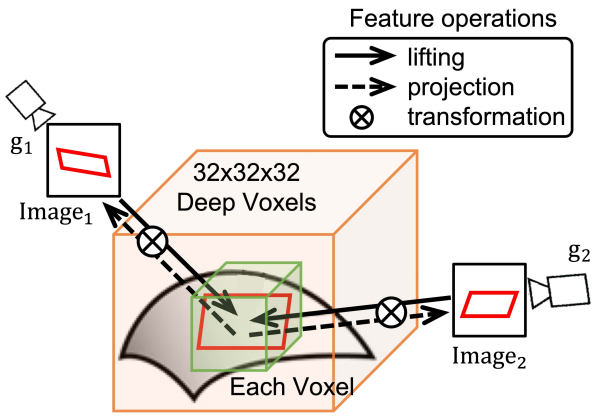


Figure 6.3: A voxel that encodes a parallelogram pattern looks different at two poses due to perspective projection effects under diffuse reflectance.

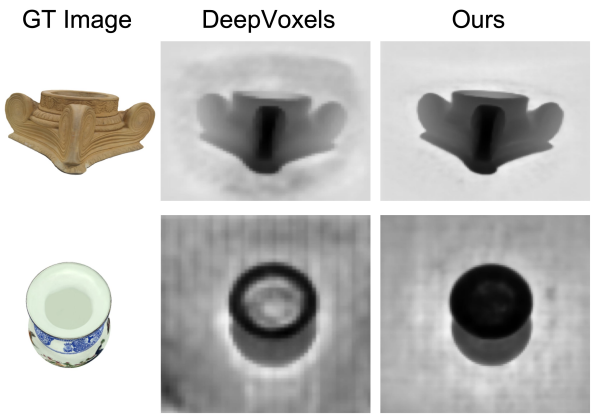


Figure 6.4: Pseudo-depth maps visualized using estimated frustum visibility values. Our results are sharper than DeepVoxels and have less artifacts.

feature patch lifting obtained voxel-shape features $X_i \in \mathbb{R}^{c\times s\times s\times s}$. Next, we project each voxel onto the feature patches $\{F_i^n\}_{n=1}^{N'}$ and conduct differentiable bilinear feature sampling to get the lifted voxel features. The projection operation, approximated via a pin-hole camera model, is also differentiable. Note that the intrinsic matrix $K \in \mathbb{R}^{3\times 3}$ *wrt. the image patches* P_i^n has to be rectified to get K_r in order to correctly map world-coordinate locations onto *the extracted feature patches* F_i^n. Because an image patch and its corresponding feature patch have different sizes.

$$K_r = \begin{bmatrix} \alpha f_x & & \alpha c_x \\ & \beta f_y & \beta c_y \\ & & 1 \end{bmatrix} \tag{6.1}$$

where (f_x, f_y, c_x, c_y) belong to K. K_r is the rectified intrinsic matrix used in voxel projection, and (α, β) are (width, height) ratios between F_i^n and P_i^n.

The voxel space used to accumulate the lifted features X_i is typically of low resolution (*e.g.* 32×32×32). Therefore, each voxel can be thought of modeling a local surface region of the object, as illustrated in Fig. 6.3. It explains the motivation for perspective projection

effect modeling by voxel feature transformations during the lifting (and projection) steps. We apply learned convolutional feature transformation kernels $A(\cdot) \in \mathbb{R}^{c\times c\times 1\times 1\times 1}$ on the lifted features X_i.

$$\bar{X}_i = A(G(g_i)) \circledast X_i \tag{6.2}$$

in which $\bar{X}_i \in \mathbb{R}^{c\times s\times s\times s}$ are the transformed features and $\circledast$ represents 3D convolution operation. As shown in Fig. 6.2, the kernel estimation network $A(\cdot)$ is implemented as several 3D convolution layers that take relative voxel-camera poses $G(g_i) \in \mathbb{R}^{6\times s\times s\times s}$ as input and estimate convolutional feature transformation kernels. The reason why $G(g_i)$ is a 3D shape tensor is because each entry of it consists of the relative voxel-camera translation and the camera pose rotation vector. Note that voxels has different relative voxel-camera translations but share the same camera rotation vector. We adopt this encoding format of relative voxel-camera poses based on empirical studies.

6.3.2 Recurrent-concurrent voxel feature aggregation

The lifted and transformed features $\bar{X}_i$ from one source image S_i only provide a single-view observation of the object at pose g_i. To learn holistic 3D voxel embeddings $Z \in \mathbb{R}^{c\times s\times s\times s}$ we need to integrate features extracted from all the training views, which have about 500 images and thus cannot be aggregated into the voxels at one time. We address this challenge by aggregating $\{\bar{X}_j^k\}_{j=1}^V$ from V different views within each iteration (indexed by k) of voxel representation updates, via both recurrent gated-fusion and concurrent max-pooling. Note that the prior methods only integrate features from a single-view into Z at each feature update iteration, and therefore suffer from single-view observation bias. Our aggregation approach provides a large surface coverage of the object during each voxel representation update and improves data utilization rate.

Recurrent gated-fusion We first use 3D-GRU [CVG14] to separately fuse each single-view

Figure 6.5: Novel-view synthesis results of DeepVoxels++ on objects with large viewpoint changes and complex shape/texture patterns. Our model is proposed for diffuse objects but we also show a few preliminary results on objects with specularities and shadows.

obtained $\bar{X}_j^k$ into the holistic 3D voxel embeddings Z^{k-1} that are obtained from the previous training iteration: $Z_j^k = \text{GRU}(\bar{X}_j^k, Z^{k-1})$. The 3D object representation Z is modeled as the hidden voxel embeddings of 3D-GRU and will be recurrently updated when more views come in. At the first round of voxel aggregation, we initialize Z^0 with zeros. However, within each step of voxel embedding update, recurrent gated-fusion only aggregates features from a single-view observation. To tackle the single-view update bias, we further utilize a multi-view based max-pooling operation upon the 3D voxel embeddings.

Concurrent max-pooling Now we need to aggregate a set of deep voxel embeddings $\{Z_j^k\}_{j=1}^V$ obtained separately by recurrent gated-fusion from $V > 1$ views. Inspired by Multi-view CNN [SMK15], we use the max-pooling operation: $Z^k = \text{Max}(Z_1^k, Z_2^k, ..., Z_V^k)$. Max$(\cdot)$ means

applying max-pooling operations along the first dimension (*i.e.* the feature channel) of the 3D voxel embeddings $Z_j^k \in \mathbb{R}^{c\times s\times s\times s}$. The obtained latent voxel representation Z^k (*i.e.* the 3D-GRU hidden embedding at the k-th iteration) will be passed into the next iteration of recurrent-concurrent voxel feature update until the end of training.

6.3.3 View-dependent patch rendering

Rendering a target image $T_{|g_m}$ from the 3D voxel embeddings $Z_{(j)}^{(k)}$[1] at any given pose g_m around the object involves three steps: view-dependent frustum feature sampling, depth dimension reduction and patch-based neural rendering.

View-dependent frustum sampling For each target camera pose g_m, we define a $d\times h\times w$ frustum space to enclose the $s\times s\times s$ cubic voxels where the volumetric embeddings Z are saved. While voxels are usually of low spatial resolution (*e.g.* 32×32×32) due to GPU size constraint, the rendering visual quality from these deep voxel embeddings can be substantially improved by sufficient frustum sampling. Namely, we utilize large 2D sampling sizes $h\times w$ (*e.g.* 128×128 vs. 32×32). The depth axis d is collapsed when rendering image patches. We found that this is a simple yet effective implementation trick for deep voxels-based high quality view synthesis. Ablation studies in the experiments section support this argument (see Fig. 6.7 and Tab. 6.3). Specifically, we can map the frustum into the voxel space by inverse-perspective projection and sample the transformed voxel features $\bar{Z} \in \mathbb{R}^{c\times s\times s\times s}$ by differentiable trilinear interpolation.

$$\bar{Z} = B(G(g_m)) \circledast Y(Z) \tag{6.3}$$

here $Y(\cdot)$ is a 3D U-Net that further refines the 3D voxel embeddings Z. As shown in Fig. 6.3, we need to conduct voxel feature transformations at both lifting and projection steps due

[1]During training gradients are back-propagated to Z_j^k. At test time we use the converged Z for rendering. We use Z for convenience from here.

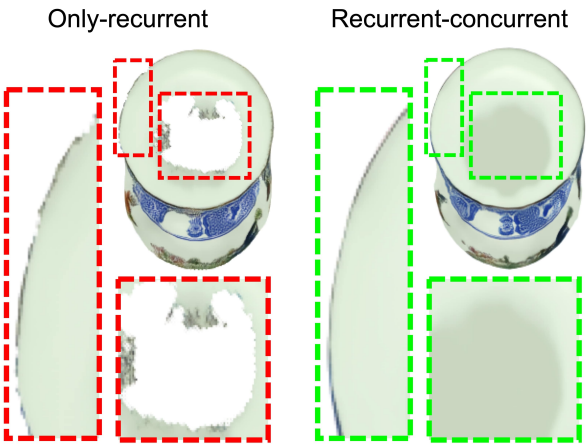


Figure 6.6: 3D voxel embeddings aggregation: only-recurrent vs. recurrent-concurrent.

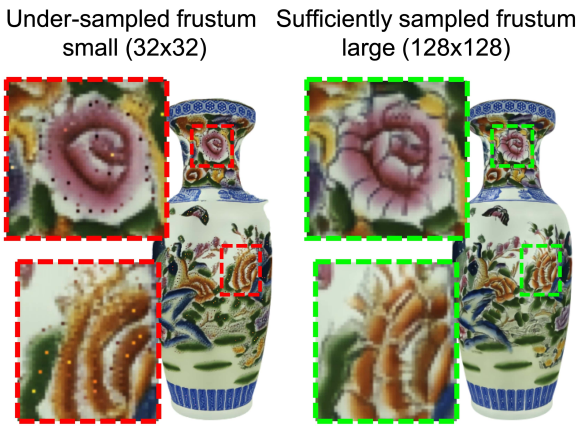


Figure 6.7: Frustum representation sampling sizes: small (32×32) vs. large (128×128).

to the corresponding perspective projection effect in the observed/rendered images. Thus, similar to Eq. 6.2, we use a kernel estimation network $B(\cdot)$ to directly take the relative voxel-camera poses $G(g_m) \in \mathbb{R}^{6\times s\times s\times s}$ as input and estimate convolutional feature transformation kernels. $B(\cdot)$ is also implemented as several 3D convolution layers. The sufficiently sampled frustum features from $\bar{Z}$ are denoted as $F_{|g_m} \in \mathbb{R}^{c\times d\times h\times w}$. Note that, as per Eq. 6.1, we use a rectified camera intrinsic matrix when conducting inverse-perspective projection for frustum representation sufficient sampling. In this case, scaling factors (α, β) are (width, height) ratios between the frustum and the target image.

Depth dimension reduction Rather than directly utilizing the frustum representation $F_{|g_m}$ for patch neural-rendering, we follow [STH19a] and first collapse it into depth dimension reduced features $H_{|g_m} \in \mathbb{R}^{c\times h\times w}$ by weighted average feature pooling upon the depth dimension: $H_{|g_m} = \text{Avg}[F_{|g_m} \otimes O(F_{|g_m})]_{|dim=1}$. Here $\text{Avg}[\cdot]_{|dim=1}$ indicates weighted average feature pooling along the second dimension (*i.e.* depth) of the $c\times d\times h\times w$ input tensor. $\otimes$ means element-wise multiplication with broadcasting between $F_{|g_m} \in \mathbb{R}^{c\times d\times h\times w}$ and $O(\cdot) \in \mathbb{R}^{1\times d\times h\times w}$. $O(\cdot)$ is implemented as a 3D U-Net with skip connections, whose output can be treated as

frustum visibility estimation *wrt.* a viewpoint g_m and adds interpretability to the rendering process. Because it enables the computation of pseudo-depth maps which explain several rendering artifacts of the prior methods (see Fig. 6.4). Specifically, inaccurate visibility estimation, induced by incorrectly up-weighting (in)visible surfaces and empty space within the frustum, can cause DeepVoxels' rendering artifacts like aliasing and holes.

Patch-based neural rendering The final step of the model is to render patches from $H_{|g_m}$. Recall that during the encoder stage, we explained the benefits of *patch-based feature extraction* (subsec. 6.3.1). During the decoding step we conduct patch-based neural rendering, for the same purposes of utilizing fewer 2D U-Net parameters, reducing the complexity of large image context modeling and being able to model/render images at high resolution. Similar to patch-based feature extraction, we subdivide $H_{|g_m}$ into small-size feature patches $\{h^n\}_{n=1}^{N}$ by a sliding window with overlaps, and then conduct patch neural rendering using a 2D U-Net with skip connections: $h^n \mapsto P^n$. At training time, we apply random sampling to retain $N^{'}$ feature patches (*e.g.* 80% of N) to save GPU memory. We use L^1 reconstruction losses upon rendered image patches to enable joint training for the complete network architectures as shown in Fig. 6.2.

$$L(\hat{P}^n, P^n) = \frac{\sum_{n=1}^{N^{'}} \sum_{a,b} \left\| \hat{P}^n_{a,b} - P^n_{a,b} \right\|_1}{N^{'} * D} \tag{6.4}$$

where $\hat{P}^n$ is a rendered image patch and P^n is a ground-truth patch. (a, b) are pixel indices within an image patch and D is the pixel number of a patch. At test time, we composit all N rendered patches $\{\hat{P}^n\}_{n=1}^{N}$ into the target image raster, and crop overlapped regions. The stitched patches comprise the final 512×512 color rendered image $\hat{T}_{|g_m}$.

6.4 Implementation Details

We implement our approach using PyTorch [PGC17]. The networks are trained with the ADAM optimizer [KB14] using an initial learning rate of 0.0004. For different benchmark

Method	Vase PSNR / SSIM	Pedestal PSNR / SSIM	Chair PSNR / SSIM	Cube PSNR / SSIM	Mean PSNR / SSIM
Nearest Neighbor	23.26 / 0.92	21.49 / 0.87	20.69 / 0.94	18.32 / 0.83	20.94 / 0.89
Tatarchenko *et al.* [TDB15]	22.28 / 0.91	23.25 / 0.89	20.22 / 0.95	19.12 / 0.84	21.22 / 0.90
Worrall *et al.* [WGT17]	23.41 / 0.92	22.70 / 0.89	19.52 / 0.94	19.23 / 0.85	21.22 / 0.90
Pix2Pix [IZZ17]	26.36 / 0.95	25.41 / 0.91	23.04 / 0.96	19.69 / 0.86	23.63 / 0.92
Neural Volumes [LSS19]	20.39 / 0.84	36.47 / **0.99**	35.15 / **0.99**	26.48 / 0.96	29.62 / 0.95
DeepVoxels [STH19a]	27.99 / 0.96	32.35 / 0.97	33.45 / **0.99**	28.42 / 0.97	30.55 / 0.97
Ours	**32.91** / **0.98**	**38.93** / 0.98	**40.87** / **0.99**	**36.51** / **0.99**	**37.31** / **0.99**

Table 6.1: 360° novel-view synthesis benchmark of objects with diffuse reflectance. Higher values of PSNR and SSIM indicate better rendering quality. Our method DeepVoxels++ surpasses DeepVoxels and other competing methods by large margins.

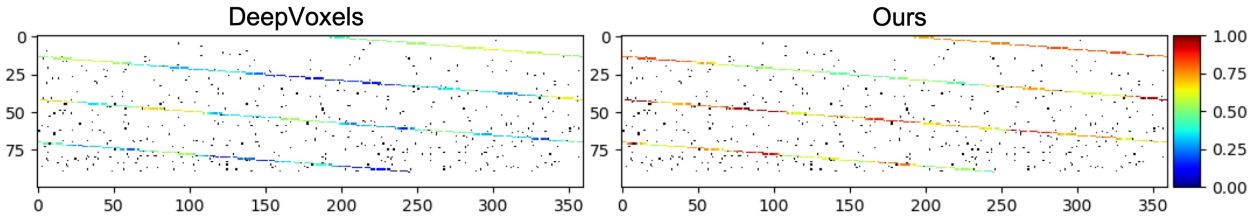


Figure 6.8: Normalized azimuth-elevation PSNR maps on Cube. Horizontal: $[0°, 360°]$ azimuth. Vertical: $[0°, 100°]$ elevation. Black dots are the training poses. Colored spiral lines are the test pose trajectories. Red color means large PSNR value and blue means small. These plots showcase smooth viewpoint interpolation paths between training views (*i.e.* black dots), showing consistent improvement over DeepVoxels across different novel views.

objects, we use the same set of hyper-parameters and stop the training at 400 epochs, which takes about 4 days.

V	Vase	Pedestal	Chair	Cube	Mean PSNR	Mean SSIM
1	27.99	32.35	33.45	28.42	30.55	0.97
4	**30.30**	34.64	35.97	**31.97**	33.22	**0.98**
8	29.45	**35.54**	**37.79**	31.65	**33.61**	**0.98**

Table 6.2: Better rendering quality can be achieved when more multi-view images V are aggregated in each round of recurrent-concurrent latent 3D voxel embedding updates.

6.5 Experiments

We evaluate DeepVoxels++ on 360° novel-view synthesis benchmarks against several competing methods: a Nearest Neighbor baseline, Tatarchenko et al. [TDB15], Worrall et al. [WGT17], Pix2Pix [IZZ17], Neural Volumes [LSS19], SRN [SZW19] and DeepVoxels [STH19a]. To add interpretability of our model, we also conduct ablation studies to reveal the impact of our series of enhancements.

6.5.1 Dataset and metrics

For fairness of comparison, we use the same dataset and evaluation metrics (*e.g.* the Structural Similarity Index (SSIM), the Peak Signal-to-noise Ratio (PSNR)) as DeepVoxels [STH19a]. The dataset contains 512×512 color images of delicate shapes and appearance (*e.g.* pedestal, vase). For each object, the dataset has about 500 training images and 1000 test views as ground truth. The test views are densely sampled from a 360° spiral curve enclosing the object at different angles and distances, for evaluating smoothness and fidelity as the viewpoint changes. This contrasts with recent object based novel-view synthesis papers which mainly use the 256×256 ShapeNet image dataset. ShapeNet lacks the aforementioned appearance complexity, and does not evaluate novel-view rendering results at densely sampled test poses. Though not the purpose of our method, we also show a few results on objects with specular reflectance and shadows in order to shed light on future work.

6.5.2 Evaluating 360° novel view synthesis

Once trained on multi-view images of an object, we no longer requires those views at *test time* as reference-view input; a requirement of some recent methods [ZTS16, PYY17, SHL18]. Rather, we can directly use the learned 3D voxel embeddings Z to render high-resolution images at novel views. Tab. 6.1 shows our method DeepVoxels++ to outperform DeepVoxels [STH19a] by 22% PSNR improvement and 33% SSIM error reduction. Both DeepVoxels and Neural Volumes [LSS19] are based on (deep) voxel representations. Our method also surpasses a recent implicit neural representation method SRN [SZW19] but it only reported mean results: 33.03 PSNR, 0.97 SSIM. We further visualize normalized azimuth-elevation PSNR maps in Fig. 6.8 to prove that our improvement is due to consistently improved rendering quality across 1000 dense test views of the object, not caused by over-fitting at certain viewpoints that are close to the training data. This capability to smoothly interpolate between training views at high fidelity contrasts with DeepVoxels.

Figs. 6.1 and 6.5 present rendering results on diverse objects. While the competing methods and our approach are proposed for and benchmarked on objects of diffuse reflectance, in the figures we also show some preliminary results on specularity and shadow modeling. Our rendered images contain sharper details and fewer rendering artifacts such as blur, aliasing and holes than DeepVoxels.

6.5.3 Ablation studies

Voxel feature aggregation Previous deep voxel methods [TCF19, STH19a] use 3D-GRU [CGC15, HS97] for image-based modeling by adopting a structured voxel space as the hidden embedding and treating hundreds of multi-view images of an object as a video sequence. However, this type of single-view based *sequential* update manner can cause inefficiency and bias of 3D voxel embeddings learning. Because it imposes an ordering on viewpoints and biases training when only a single-view observation is aggregated during each recurrent

Sampling sizes $h \times w$	Vase	Pedestal	Chair	Cube	Mean PSNR	Mean SSIM
32×32	27.16	27.93	32.99	27.35	28.86	0.95
64×64	30.30	34.64	35.97	31.97	33.22	0.98
128×128	**32.62**	**38.75**	**38.73**	**35.35**	**36.36**	**0.99**

Table 6.3: Frustum representation sufficient sampling from the low spatial resolution deep voxels can substantially improve the 360° novel-view synthesis performance.

step. Inspired by Multi-view CNN [SMK15], we address these challenges by conducting voxel feature aggregation at two different dimensions jointly: recurrent gated-fusion and concurrent max-pooling. This provides a large surface coverage of the object during each iteration of voxel feature updates and improves data utilization rate. Results in Tab. 6.2 verify our arguments. Fig. 6.6 further demonstrates that our aggregation method helps to reduce DeepVoxels' rendering artifacts such as aliasing and holes. Better rendering quality and faster training can be achieved with more views aggregated by max-pooling in each round of voxel feature updates, which is most effective when view number increases from 1 to 4 and starts to become less effective when it reaches 8 views. Thus, in our benchmark results we use 4 views considering the trade-off between performance gains and GPU memory size constraints.

Frustum sufficient sampling To decode the learned 3D voxel embeddings and render an image at a target pose g_m, we need to first project the deep voxel features into a frustum. As explained in subsec. 6.3.3 the projection procedure essentially is feature sampling from the voxel space to the frustum space. Tab. 6.3 and Fig. 6.7 show that while voxels are usually of 32×32×32 low spatial resolution due to GPU memory constraints, sufficient frustum sampling can substantially improve the visual quality of rendered images with sharper details than DeepVoxels. The frustum representation sampling sizes are determined by height/width of the depth dimension reduced frustum feature maps. We use 128×128 sampling sizes in our main results. Frustum representation sufficient sampling is *a simple yet effective implementation*

		2D U-Net parameters (M)	
Method	Mean PSNR	Feature extraction	Neural rendering
Full-image	36.36	92.2	108.9
Patch	**36.99**	**40.3**	**56.9**

Table 6.4: Our patch-based scheme reduces the complexity of large image context modeling and improves the rendering results by learning local shape patterns.

Method	Vase	Pedestal	Chair	Cube	Mean PSNR	Mean SSIM
Without	**26.05**	29.84	28.89	25.19	27.49	0.93
With	25.76	**30.83**	**29.45**	**25.43**	**27.87**	**0.94**

Table 6.5: Comparisons between without/with voxel feature transformations. With the learned feature transformation kernels, we achieve better performance on objects of delicate shapes (*e.g.* pedestal, chair) and limited training views (*e.g.* 30 images).

trick that addresses the low voxel resolution problem. One explanation is that though voxels have low spatial resolution, they contain high dimensional latent 3D embeddings, encoding objects' appearance and shape information. Meanwhile, the differentiable trilinear interpolation-based frustum sufficient sampling enforces strong supervision on the deep voxel features (*i.e.* rich gradient signals), and eventually helps to encode more fine-scale details into the learned latent 3D embeddings.

Low-complexity patch modeling The patch-based training/inference scheme has multiple advantages over the previous full-image based one, which are also demonstrated in other problems like point-cloud upsampling [YLF18], image restoration [PE15]. Besides reducing the complexity of modeling large image context and therefore improving fine-scale patch synthesis quality (see Tab. 6.4), our approach requires only half the 2D U-Net parameters used in image feature extraction and neural rendering of prior methods due to small-size input. Furthermore, the patch-based scheme enables us to model and render images of

high resolution at low GPU memory cost, whereas full-image based methods are not easily trainable. Though small patch sizes enable the network to reduce its receptive field and capture local shape patterns, too small sizes can harm the learning and rendering results due to lacking sufficient image context. Therefore, there is a sweet spot of patch sizes. We tested 3 different patch sizes on Cube (512x512, 128x128, 64x64) and the PSNR/SSIM are: 35.35/0.992, 36.27/0.993, 32.08/0.980. We kept the 128x128 patch size as our default configuration.

View-dependent voxel feature transformation Fig. 6.3 illustrates how 3D voxel embeddings that encode shape and appearance of an object's local surface plane can be mapped to different patterns, due to the corresponding perspective projection effect induced by viewpoint changes. Such perspective transformation in the observed/rendered images is explicitly modeled via learned feature transformation kernels from relative voxel-camera poses. In contrast, previous methods rely on voxel volume changes caused by vantage point changes to infer view-dependency. But voxel volume differences are constrained by low voxel spatial resolutions and only implicitly reflect viewpoints. Tab. 6.5 shows that explicit voxel feature transformation modeling is critical for objects with delicate shapes (*e.g.* pedestal) and limited training views (*e.g.* 30 images), where voxel volume changes are less continuous and less effective to model the corresponding perspective transformation caused by viewpoint changes under diffuse reflectance.

Number of training views While DeepVoxels requires around 500 multi-view images to learn faithful 3D voxel embeddings of an object, DeepVoxels++ can learn to synthesize high fidelity novel views even with a limited number of training images. In Tab. 6.6, we experiment on full-size, 1/3, 1/16 and 1/48 training data. Our method outperforms DeepVoxels in all conditions, demonstrating promising results for real-world applications where only few images are available for 3D object representation learning and novel view synthesis. For example, camera rig-based image capture systems.

	Training data sizes			
Method	full	1/3	1/16	1/48
DeepVoxels	30.55	28.09	26.06	19.35
Ours	**37.31**	**33.34**	**27.87**	**20.71**

Table 6.6: Our model surpasses DeepVoxels under different data size configurations by large gaps in mean PSNR. The results indicate that DeepVoxels++ is data efficient.

6.6 Discussion

We have proposed a novel view modeling and rendering technique that learns latent 3D voxel embeddings from multi-view images of an object without 3D occupancy supervision. Our approach outperforms previous deep voxel-based methods by large margins on 360° novel-view synthesis benchmarks. We show that our results contain more fine-scale details and less rendering artifacts than DeepVoxels [STH19a]. We also conduct multiple ablation studies to show the impact of our series of improvements in achieving this enhanced rendering fidelity.

Although the benchmark mainly evaluates objects with diffuse reflectance, our proposed method of learning voxel feature transformation kernels potentially can also model other view-dependent effects (*e.g.* specularity) besides image-plane perspective transformations of diffuse surfaces. We demonstrate some preliminary visual results for specularity and shadow modeling in Fig. 6.5. But it is worth considering extending the current dataset with objects of non-Lambertian reflectance and conducting evaluations under various lighting situations. Novel-view rendering for non-rigid objects leveraging dynamic volumes is another challenging and important problem. In brief, future work could consider various scenarios that are not explicitly modeled or extensively evaluated by the current deep voxel-based methods, such as lighting and specular reflectance, multi-object scenes, dynamic objects and so on.

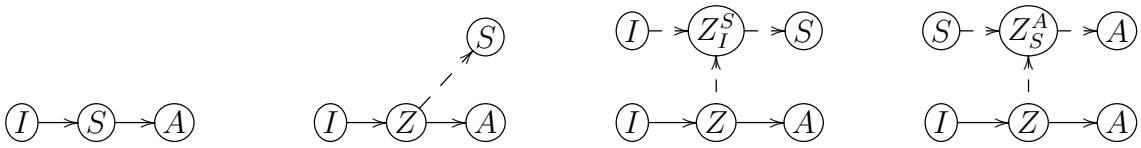
CHAPTER 7

End-to-End Visual Driving Policy Learning

7.1 Introduction

Driving is a complex endeavor that consists of many sub-tasks such as lane following, making turns, and stopping for obstacles. A traditional strategy to tackle this complexity is to split the driving policy into two stages: perception (i.e. estimating a manually chosen representation of the scene) and control (i.e. outputting low-level controls using hand-coded rules). Though this approach is interpretable with easy-to-diagnose failures, it suffers from various issues: the representations may be suboptimal and difficult to estimate while hand-coded rules may struggle to handle the full range of driving scenarios [SSS16, LWY18, SSG18]. These drawbacks motivate academia to explore learning approaches for driving as they are not restricted by hand-coded decisions. A major challenge for learning approaches is obtaining a useful representation. A simple approach is to directly learn a driving model that maps from image to low-level control. Though this approach is easy to implement, it suffers from poor generalization due to overfitting to nuisances [LMS18, CSL19].

To improve generalization, various approaches (Fig. 7.1) propose to leverage complex side tasks such as semantic segmentation which provide driving-relevant contextual knowledge in an easily accessible form (i.e. without photometric nuisances). The most straightforward of these approaches is the two-stage approach [MDG18, HXS19, BCP20, BKO19] (Fig. 7.1a), which learns to achieve the side task and then uses the output from the side task to estimate controls. However, this method suffers from errors in directly estimating the complex side



(a) Two-stage **(b)** Multi-task **(c)** Feature mimic **(d)** Ours

Figure 7.1: Approaches for leveraging side tasks for driving. I, S, and A represent the image, side task annotations, and low-level controls; Z, Z_I^S, and Z_S^A, latent representation, mimicked features, and the squeezed encoding of the side task annotations. Dashed arrows, depending on direction, represent branches or supervision used only during training. Unlike other methods that aim to achieve the side task (two-stage, multi-task) or learn features for it (feature mimic), our method, SAM, learns only driving-relevant context from the side task annotations by directly feeding them as input during training. Also, note that SAM does not suffer from side task estimation errors encountered by other methods because we never explicitly estimate any side task annotations during training/inference.

task, leading to unrecoverable failures in the driving policy.

To avoid these estimation errors at test time, multi-task learning approaches [XGY16, LMS18] (Fig. 7.1b) do not estimate the side task at test time but instead incorporate driving-relevant context by training a shared representation to achieve both the side task and the main driving task. However, they suffer from the fact that the side task is distinct from the main task (i.e. side task-main task mismatch), and hence, there exists information relevant to the side task but not to the main task. For example, the side task of semantic segmentation encourages the learned representation to be informative for every pixel's class label, but knowing such pixel-wise information is unnecessary for driving. Instead, what is more important is knowing that an obstacle of a certain class is in front of the agent. Hence, as multi-task approaches aim to achieve both the side task and the driving task, they encourage the representation to contain side task-relevant but driving-irrelevant nuisances, hurting performance.

Feature mimicking [HML] (Fig. 7.1c) incorporates contextual information while reducing the reliance on ground-truth annotations for the side tasks (side task annotations) by proposing instead to mimic features of a network (pre)trained to achieve the side task. However, this method suffers from a similar issue (side task-main task mismatch) to multi-task methods: as feature mimicking encourages the representation, via the mimicked features, to be informative for achieving the side task, it encourages the learning of side task relevant but driving-irrelevant nuisances. Moreover, all these previous methods (Fig. 7.1) inevitably suffer from side-task estimation errors during training/inference as they rely on training models to explicitly estimate the side task.

In contrast with previous approaches, we propose a method (Fig. 7.1d) that effectively leverages the side task for conditional driving policy learning without trying to achieve the side task or learn features trained to do so. Instead, we aim to learn a representation of the side task annotations that is directly useful for driving. We first train a squeeze network to drive using ground-truth side task annotations as input and extract its deep features. As these features are trained only for driving and not to achieve the side task, they contain only driving-relevant knowledge and not the side task-relevant but driving-irrelevant nuisances. However, we cannot use the squeeze network to drive as ground-truth side task annotations are unavailable at test time. To train a driving policy that does not require these annotations while still leveraging the side task, we train a mimic network to drive using only image and self-speed as input while pushing it to learn the squeeze network's embedding via a mimicking loss. Hence, the mimicking loss encourages the mimic network's representation to learn only the driving-relevant context associated with the side task. Our overall approach, squeeze-and-mimic networks (SAM), leverages the side task in an effective way to train a conditional driving policy without achieving the side task or learning features for it, avoiding the pitfalls of previous methods such as estimation errors and learning side task relevant but driving-irrelevant nuisances.

We note that for all the aforementioned methods (Fig. 7.1), the side tasks should not

discard important driving-relevant information contained in the image input. Indeed, they should contain this information in an easily accessible form (i.e. without photometric nuisances) so that the intermediate representation can easily learn this information. Two important yet complementary types of information are object class, useful for tasks like lane following, and braking behavior. Hence, for our choice of side task annotations, we select semantic segmentation and *stop intention values,* which indicate whether to stop for different types of obstacles such as pedestrians, traffic lights, or cars.

We test our agent on online benchmarks using the photorealistic CARLA simulator [**?**]. Furthermore, we find previous benchmarks lacking as we do not think one should be rewarded for reaching the destination while driving through red lights or into the opposite lanes. Therefore we introduce a more realistic evaluation protocol that rewards successful routes only if they do not violate basic traffic rules. To summarize, our main contributions are as follows:

- a method for conditional driving policy learning that leverages a side task and learns only driving-relevant knowledge from its annotations, avoiding the pitfalls of existing methods that aim to achieve the side task or learn features for it.

- the combination of complementary side tasks: semantic segmentation, which provides basic class concepts for subtasks like lane following, and stop intentions, which provide causal information linking braking to hazardous driving scenarios. While both tasks lead to improvement over the baseline, we show that the best performance is achieved only when they are used jointly.

- an evaluation protocol, Traffic-school, that fixes the flaws of prior benchmarks. Though CARLA reports various driving infractions, these statistics are scattered and hard to analyze. Thus, previous benchmarks usually compute the route success rate, but they ignore several risky driving behaviors. In contrast, our protocol sets a high standard for the autonomous agent, penalizing previously ignored infractions such as violating

red lights and running into the sidewalk.

7.2 Related Work

Autonomous driving has drawn significant attention for several decades [Pom88, LMB05, SBS10]. Generally, there exist three different approaches to driving: modular, direct perception, and end-to-end learning.

Modular pipelines form the most popular approach [Ull80, PCY16, Fra17, DC91], separating driving into two components: perception modules [GLS13, HS19, **?**] that estimate driving-related side tasks and control modules. Recently, [MDG18, HXS19, BCP20, BKO19] use semantic segmentation and environment maps as side task annotations in a modular pipeline. However, these annotations are complex and high-dimensional. To simplify the annotations and alleviate the annotation burden, a recent work LEVA [BCP20] proposes to use coarse segmentation masks. The mentioned modular approaches have the advantage of outputting interpretable representations in the form of estimated side task annotations (perception outputs), allowing for the diagnosis of failures. However, they suffer from perception and downstream control errors, *and* use manually chosen representations that are potentially suboptimal for driving.

Direct perception [Gib79] methods similarly separate the driving model into two parts, but unlike modular approaches, they avoid complex representations, opting for compact intermediate representations instead. [CSK15] and [SSG18] (CAL) both train a network to estimate affordances (distance to vehicle, center-of-lane, etc.) linked to controls, for car racing and urban driving, respectively. Similar to modular approaches, the intermediate representation is hand-crafted, with no guarantees that it is optimal.

End-to-end methods [Pom88, LMB05, SBS10, BTD16] learn a direct mapping from image to control and are most related to our work. CIL [CML18] builds upon offline behavioral cloning, solving the ambiguity problem at traffic intersections by conditioning on turning

commands. [CSL19] analyzes issues within the CIL approach and proposes an improved version, CILRS. However, both methods fail to generalize to dense traffic and suffer from the covariate shift between offline training data and closed-loop evaluation. To reduce this covariate shift, various methods collect data online using DAgger [RGB11] imitation learning [CZK19, PBO20] or reinforcement learning (RL) [YPW17, LWY18, TWM19, OPB20]. Recently, [TWM19] combines RL with a feature extractor pretrained on affordance estimation while LSD [OPB20] combines RL with a multimodal driving agent trained via mixture of experts. However, the above methods all use online training, which is expensive and unsafe and can be performed only in simulation [**?**]. Due to these drawbacks, we focus on offline methods as offline and online methods are not directly comparable.

Previous methods have also leveraged side task annotations to improve generalization and acquire context knowledge. As discussed in the Introduction and shown in Fig. 7.1, the two-stage, multi-task and feature mimic methods [BCP20, XGY16, LMS18, HML] all suffer from side task estimation errors and tend to learn driving-irrelevant nuisances (side task-main task mismatch). We overcome these issues by never aiming to achieve the side task; instead, we mimic a representation squeezed from side task annotations and trained for driving. Similar to our method, LBC [CZK19] also mimics a driving agent. However, this agent, unlike ours, takes expensive 3D side task annotations as input. Moreover, LBC and SAM use different methods of mimicking: LBC mimics the agent's *controls* while SAM mimics the agent's *embeddings*. Hence, our method directly encourages the model's representation to contain driving-related context associated with the side task while LBC does not. To illustrate the benefits of our mimicking method, we also test SAM using LBC's control mimicking.

7.3 Methodology

The overall pipeline of our method *SAM* (squeeze-and-mimic networks) is shown in Fig. 7.2. The goal of conditional driving policy learning is to learn a policy $(I, m, c) \mapsto a$

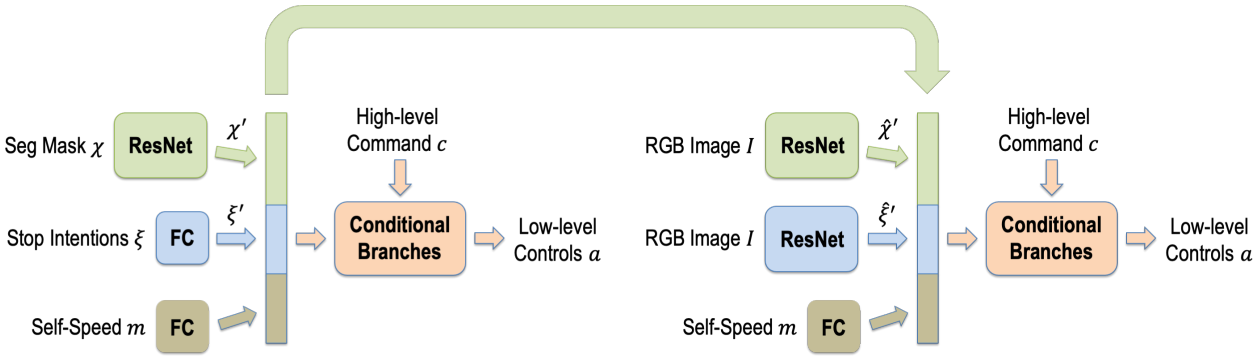


Figure 7.2: Architecture overview of our squeeze and mimic networks. Both use a three-branch structure. (Left) Squeeze network: processes semantic segmentation, stop intention values, and self-speed in separate branches and feeds their concatenated representation to a command-conditioned driving module. (Right) Mimic network: has a similar structure but takes in only RGB image and self-speed. (Arrow) l_2 loss is enforced between the segmentation and intentions embeddings of the squeeze network and those of the ResNet branches, which each take in image, of the mimic network.

that outputs low-level controls a given image I, self-speed m, and turning command $c = \{\text{follow, left, right, straight}\}$. Low-level continuous controls $a = (b, g, s)$ consist of brake b, gas g, and steering angle s.

For our method, we additionally leverage side task annotations $S = (\chi, \xi)$, in particular semantic segmentation χ and stop intention values ξ provided by the CARLA simulator, to train a conditional driving policy. The stop intentions $\xi = (v, p, l) \in [0, 1]^3$ indicate how urgent it is for the agent to brake in order to avoid hazardous traffic situations such as collision with vehicles, collision with pedestrians, and red light violations, respectively. They are like instructions given by a traffic-school instructor, which inform you of the causal relationships between braking and different types of dangerous driving scenarios, complementing the semantic segmentation χ, which contains concepts of object class identity for tasks like lane following. Hence, the training dataset for our method consists of temporal tuples $\{I_t, m_t, c_t, \chi_t, \xi_t, a_t\}_{t=1}^{T}$, collected from a rule-based autonomous driving agent with access to

all internal state of the CARLA driving simulator.

In our method *SAM*, we first squeeze the side task annotations S_t, consisting of segmentation masks χ_t and three-category stop intentions $\xi_t = (v_t, p_t, l_t)$, into a latent representation $(\chi_t^{'}, \xi_t^{'})$ containing driving-relevant info and not driving-irrelevant nuisances by training a squeeze network to drive using side task annotations, self-speed, and turning command as input: $(S_t, m_t, c_t) \mapsto a_t$. We then train a mimic network $(I_t, m_t, c_t) \mapsto a_t$ to drive with no access to side task annotations while encouraging this network's embedding $(\hat{\chi}_t^{'}, \hat{\xi}_t^{'})$ to mimic the squeeze network's embedding. Notably, the squeeze network does not take image as input; this is so that its latent representation, which is used to supervise the mimic network, does not contain photometric nuisances from the image.

7.3.1 Squeeze Network

The task of our squeeze network is to squeeze the ground-truth segmentation masks χ_t and three-category stop intentions $\xi_t = (v_t, p_t, l_t)$ into a representation that contains contextual driving information but with nuisances removed. Such a representation will later be used to supervise the mimic network via a mimicking loss. As shown in Fig. 7.2, the squeeze network uses a three-branch architecture for estimating controls $a_t = (b_t, g_t, s_t)$. The first branch uses a ResNet34 backbone to squeeze the segmentation mask input, which provides concepts of object class useful for tasks like lane following. We use the middle fully-connected (FC) branch to process the three-category stop intentions, which complement the segmentation mask by informing the agent of the causal relationships between braking behaviors and the presence of objects in the context of safe driving. The lower FC branch ingests self-speed, which provides context for speeding up / slowing down [CSL19].

The latent feature vectors from the three branches are concatenated and fed into a driving module, composed of several FC layers and a conditional switch [CML18] that chooses one out of four different output branches depending on the given turning signals c_t. The four output branches share the same network architecture but with separately learned weights.

We use ℓ_1 losses for training:

$$L_{control} = \lambda_1|\hat{b}_t - b_t| + \lambda_2|\hat{g}_t - g_t| + \lambda_3|\hat{s}_t - s_t| \tag{7.1}$$

where $(\hat{b}_t, \hat{g}_t, \hat{s}_t)$ are estimated controls of brake, throttle, steering angle respectively, and (b_t, g_t, s_t) are ground-truth controls. λ_i's are weights for loss terms.

7.3.2 Mimic Network

The mimic network does not have direct access to side task annotations, but instead observes a single RGB image I_t, self-speed measurement m_t, and high-level turning command c_t and mimics the squeeze network's latent embedding to learn driving-relevant context without learning driving-irrelevant nuisances. Its goal is also to estimate low-level controls a_t, for which it also adopts a three-branch network. The first and the second branches are now both ResNet34 backbones, pretrained on ImageNet, that separately take in image I_t and output latent embeddings $(\hat{\chi}_t^{'}, \hat{\xi}_t^{'})$. We adopt separate branches as this separation aids mimicking the embeddings of the segmentation masks, which are pixel-wise and provide object class identity, and of the intention values, which are global and provide causal links between braking and dangerous situations, respectively.

When training the mimic network, as illustrated in Fig. 7.2, besides applying the ℓ_1 control loss (Eq. (7.1)), we enforce ℓ_2 mimicking losses with weights $\lambda_i^{'}$ to encourage the mimic network's embeddings $(\hat{\chi}_t^{'}, \hat{\xi}_t^{'})$ to mimic the squeeze network's embeddings $(\chi_t^{'}, \xi_t^{'})$ of the side task annotations:

$$L_{mimic} = \lambda_1^{'} \left\| \hat{\chi}_t^{'} - \chi_t^{'} \right\|_2^2 + \lambda_2^{'} \left\| \hat{\xi}_t^{'} - \xi_t^{'} \right\|_2^2. \tag{7.2}$$

The mimicking and control losses are combined to learn a driving policy: $L = L_{control} + L_{mimic}$.

7.4 Implementation Details

We implement our approach using CARLA 0.8.4 [**?**]. To train both the squeeze and mimic networks, we use a batch size of 120 and the ADAM optimizer with initial learning rate 2e-4. We use an adaptive learning rate schedule, which reduces the learning rate by a factor of 10 if the training loss has not decreased in 1000 iterations. We use a validation set for early stopping, validating every 20K iterations and stopping training when the validation loss stops decreasing.

Regarding time complexity, our mimic network demonstrates real-time performance (59 FPS) on a Nvidia GTX 1080Ti. Training the squeeze network takes 10 hours on a Nvidia GTX 1080Ti while the mimic network trains in 1 day on a Nvidia Titan Xp.

7.5 Experiments

We demonstrate the effectiveness and generalization ability of our method by evaluating it on standard closed-loop evaluation benchmarks. In section 4.1, we compare our model against a suite of competing approaches. Concerned about the fact that multiple driving infractions are not penalized in existing benchmarks, we then introduce a more realistic new evaluation standard *Traffic-school* in section 4.2. In addition, in sections 4.3 and 4.4, we conduct ablation studies on the strategy for leveraging the side task and individual types of side task annotations. For all benchmarks, we evaluate in four combinations of towns, which differ in road layout and visual nuisances, and weathers to test generalization: training conditions, new weather, new town, and new town/weather.

7.5.1 Comparison with State-of-the-Art on CARLA

We first test our method on the NoCrash and Traffic-light benchmarks [CSL19]. For NoCrash, we also include, for completeness, results for LSD [OPB20], LBC [CZK19], and LEVA [BCP20]

even though they are not directly comparable to our method. LSD [OPB20] uses online training, which is unsafe, and hence, is dependent on a driving simulator, while our method uses only offline training, avoiding these drawbacks. LBC [CZK19] also uses online training and in addition uses expensive ground-truth 3D maps, collecting which requires "accessing the ground-truth state of the environment," which "is difficult in the physical world" [CZK19]. In contrast, our method uses only 2D segmentation and stop intentions. Finally, LEVA uses different pretraining (using MS-COCO segmentation dataset) and a significantly larger architecture (∼100M parameters for LEVA vs. ∼45M for our model). For fair comparisons, in Tab. 7.1, we also test LEVA (see "efficient seg") and LBC (see "control mimic") using comparable backbones and the same dataset as SAM.

NoCrash We report results on the NoCrash benchmark in Tab. 7.1, where the metric is navigation success rate in three different levels of traffic: empty, regular, and dense. A route is considered successful for NoCrash only if the agent reaches the destination within the time window without crashing. Though the *second-best CILRS* outperforms the other comparable baselines, our method, with average error **28%**, still achieves a relative failure rate reduction of **30%** over CILRS, which has average error **40%**. In particular, our method achieves especially large performance gains over CILRS in new town and under regular and dense traffic. These gains demonstrate the effectiveness of both our choice of side task annotations (semantic segmentation for basic class concepts and stop intentions to aid braking) and our method of leveraging them. We also note that our method outperforms MT [LMS18] (a multi-task method), control mimic, and efficient seg. This comparison illustrates the advantages of our method over other methods of using the side task annotations; our method learns only driving-relevant context associated with the side task. On the other hand, multi-task learning, efficient seg, and control mimic suffer from learning driving-irrelevant nuisances, perception errors, and not directly imbuing the representation with contextual knowledge, respectively.

Traffic-light Results are shown in Tab. 7.2. As NoCrash does not directly penalize running red lights, [CSL19] proposes the Traffic-light benchmark to analyze traffic light violation

Method	Training			New weather			New town			New town/weather			Mean
	Empty	Regular	Dense	Empty	Regular	Dense	Empty	Regular	Dense	Empty	Regular	Dense	
Comparable Baselines													
CIL [CML18]	79 ± 1	60 ± 1	21 ± 2	83 ± 2	55 ± 5	13 ± 4	48 ± 3	27 ± 1	10 ± 2	24 ± 1	13 ± 2	2 ± 0	36 ± 0.7
CAL [SSG18]	81 ± 1	73 ± 2	42 ± 3	85 ± 2	68 ± 5	33 ± 2	36 ± 6	26 ± 2	9 ± 1	25 ± 3	14 ± 2	10 ± 0	42 ± 0.8
MT [LMS18]	84 ± 1	54 ± 2	13 ± 4	58 ± 2	40 ± 6	7 ± 2	41 ± 3	22 ± 0	7 ± 1	57 ± 0	32 ± 2	14 ± 2	36 ± 0.8
Efficient seg* [BCP20]	**100 ± 0**	82 ± 1	38 ± 5	80 ± 2	72 ± 3	23 ± 3	91 ± 1	58 ± 3	15 ± 4	68 ± 3	47 ± 5	19 ± 6	58 ± 1.0
Control mimic* [CZK19]	**100 ± 0**	84 ± 1	31 ± 3	99 ± 1	76 ± 6	27 ± 6	84 ± 1	47 ± 2	11 ± 3	**83 ± 3**	51 ± 1	10 ± 2	59 ± 0.9
CILRS [CSL19]	97 ± 2	83 ± 0	42 ± 2	96 ± 1	77 ± 1	39 ± 5	66 ± 2	49 ± 5	23 ± 1	66 ± 2	56 ± 2	24 ± 8	60 ± 1.0
SAM	**100 ± 0**	**94 ± 2**	**54 ± 3**	**100 ± 0**	**89 ± 3**	**47 ± 5**	**92 ± 1**	**74 ± 2**	**29 ± 3**	**83 ± 1**	**68 ± 7**	**29 ± 2**	**72 ± 0.9**
Listed for Completeness													
LSD [OPB20]	N/A	N/A	N/A	N/A	N/A	N/A	94 ± 1	68 ± 2	30 ± 4	95 ± 1	65 ± 4	32 ± 3	N/A
LEVA [BCP20]	N/A	N/A	N/A	N/A	N/A	N/A	87 ± 1	82 ± 1	41 ± 1	79 ± 1	71 ± 1	32 ± 5	N/A
LBC [CZK19]	*100 ± 0*	*99 ± 1*	*95 ± 2*	*100 ± 0*	*99 ± 1*	*97 ± 2*	*100 ± 0*	*96 ± 5*	*89 ± 1*	*100 ± 2*	*94 ± 4*	*85 ± 1*	*96 ± 0.6*

Table 7.1: Results on NoCrash [CSL19] benchmark. Empty, regular and dense refer to three levels of traffic. We show navigation success rate (%) in different test conditions. Due to simulator randomness, all methods are evaluated 3 times. Bold indicates best among comparable methods. Italics indicate best among all methods, including those whose results are provided solely for completeness such as LSD [OPB20], LEVA [BCP20], and LBC [CZK19]. Note that LSD [OPB20] and LEVA [BCP20] report results only on the new town. We also include comparable baselines (marked with *), efficient seg and control mimic, for LEVA [BCP20] and LBC [CZK19]. The average error for *our model SAM* is **28%**, which is **30%** better than *the next-best CILRS*, **40%**, in terms of relative failure reduction.

Method	Train Town		New Town		Mean
	Train weather	New weather	Train weather	New weather	
CILRS [CSL19]	59 ± 2	32 ± 1	43 ± 1	35 ± 2	42 ± 0.8
SAM	**97 ± 0**	**96 ± 1**	**81 ± 1**	**73 ± 1**	**87 ± 0.4**

Table 7.2: Traffic light success rate (percentage of not running the *red* light). We compare with the best comparable baseline CILRS.

behavior using the NoCrash empty setting (no dynamic agents). The metric is traffic-light success rate, i.e. the percentage of times the agent crosses a traffic-light on green. The traffic light success rate of *our model SAM* (**87%**) is more than twice as high as *CILRS* (**42%**). Our

Method	Training			New weather			New town			New town/weather			Mean
	Empty	Regular	Dense	Empty	Regular	Dense	Empty	Regular	Dense	Empty	Regular	Dense	
CILRS [CSL19]	11 ± 2	12 ± 1	13 ± 2	2 ± 2	7 ± 2	7 ± 1	2 ± 1	7 ± 1	4 ± 1	0 ± 0	7 ± 1	2 ± 2	6 ± 0.4
SAM	$\mathbf{90 \pm 2}$	$\mathbf{79 \pm 1}$	$\mathbf{43 \pm 5}$	$\mathbf{83 \pm 3}$	$\mathbf{73 \pm 1}$	$\mathbf{39 \pm 4}$	$\mathbf{46 \pm 2}$	$\mathbf{39 \pm 3}$	$\mathbf{12 \pm 2}$	$\mathbf{15 \pm 3}$	$\mathbf{25 \pm 2}$	$\mathbf{14 \pm 0}$	$\mathbf{47 \pm 0.8}$

Table 7.3: The newly proposed ***Traffic-school*** benchmark provides a more solid evaluation standard than both the old CARLA and NoCrash, Tab. 7.1, benchmarks, which are flawed due to not penalizing infractions such as red light or out-of-road violations. On our new benchmark, a route is considered successful only if the agent arrives at the destination within a given time without a) crashing, b) traffic light violation, c) out-of-road infraction. Under this more realistic evaluation protocol, our results, in all conditions, surpass the best comparable baseline CILRS.

improved traffic light performance demonstrates the effectiveness of both our stop intentions and our method of leveraging them: the stop intentions inform the agent to stop for red lights while our method learns the context associated with them.

7.5.2 A More Realistic Evaluation Protocol: Traffic-school

To resolve the flaws of previous benchmarks, we propose the *Traffic-school* benchmark, which shares the same routes and weathers as NoCrash with a more restrictive evaluation protocol. In previous benchmarks, multiple driving infractions such as red light violations (ignored by NoCrash) and out-of-lane violations are ignored when judging whether a route is successfully finished. In the *Traffic-school* benchmark, we do not ignore such infractions; a route is considered a success only if the agent reaches the destination while satisfying the following requirements: a) no overtime, b) no crashes, c) no traffic light violation, d) no running into the opposite lane or sidewalk. As shown in Tab. 7.3, under this more realistic evaluation protocol, our results (**47%**) outperform the best comparable baseline CILRS (**6%**), showing that our method learns the driving-related context and not the nuisances associated with side tasks. In particular, effectively leveraging the semantic segmentation and stop intentions boosts

our performance for staying in the lane and stopping for red lights (Tab. 7.2), infractions previously ignored but tested by *Traffic-school*.

7.5.3 Ablation Studies about Squeeze-and-Mimic Networks

To demonstrate the effectiveness of our method of using side task annotations, we conduct an ablation study on different methods of leveraging side tasks for driving (Tab. 7.4). We compare against alternative methods for driving policy learning [MDG18, BKO19, XGY16, LMS18, HML], depicted in Fig. 7.1, using the same side tasks and comparable backbones. We also compare against baselines that do not use side tasks.

Two-stage-(F) We apply an intuitive strategy (Fig. 7.1a) of utilizing two separately trained modules: a) perception networks, b) driving networks. The perception networks are trained for segmentation masks and stop intention values estimation. In the second step, the driving networks use the same architecture as the squeeze network and take estimated segmentation masks and stop intentions as input for low-level controls estimation. For two-stage, we directly take the learned weights from the squeeze network as the driving network. Note that the squeeze network is trained with ground-truth segmentation masks and stop intention values. Thus, for two-stage-F, we fine-tune the driving network on the estimated segmentation masks and stop intentions. Using either variant, we note that this strategy suffers from perception errors, which cannot be recovered from.

Multi-task We apply a similar multi-task training strategy (Fig. 7.1b) as [XGY16] and MT [LMS18] but with our side tasks. On the same latent feature vectors where we enforce mimicking losses in our SAM method, we now train decoders to estimate segmentation masks and stop intentions as side tasks. The motivation is that by simultaneously supervising these side tasks, the learned features contain driving-relevant context such as lane markings and other agents and are more invariant to environmental changes like buildings, weather, etc. However, the downside of this side task supervision is that it encourages the learned features to contain side task-relevant but driving-irrelevant nuisances.

Method	Training weather			New weather			Mean
	Empty	Regular	Dense	Empty	Regular	Dense	
SAM-NM	65 ± 3	36 ± 1	9 ± 2	42 ± 3	31 ± 2	7 ± 3	31.7 ± 1.0
Res101-NM	70 ± 3	44 ± 2	13 ± 4	50 ± 2	33 ± 1	7 ± 3	36.2 ± 1.1
Two-stage	**92 ± 1**	50 ± 3	12 ± 1	81 ± 2	41 ± 6	9 ± 3	47.5 ± 1.3
Two-stage-F	90 ± 2	57 ± 4	13 ± 1	79 ± 3	42 ± 4	8 ± 2	48.2 ± 1.2
Feature mimic	90 ± 1	62 ± 2	18 ± 1	79 ± 2	58 ± 2	15 ± 5	53.7 ± 1.0
Multi-task	91 ± 0	62 ± 2	17 ± 2	**83 ± 1**	65 ± 6	16 ± 2	55.7 ± 1.2
SAM	**92 ± 1**	**74 ± 2**	**29 ± 3**	**83 ± 1**	**68 ± 7**	**29 ± 2**	**62.5 ± 1.4**

Table 7.4: Comparison of alternative methods that leverage the side task on NoCrash. We show navigation success rate in the new town. Though two-stage-(F), multi-task, and feature mimic improve over the aforementioned non-distillation models by large gaps, they still perform worse than our SAM model, showing that among multiple alternatives of using the segmentation masks and stop intention values, our method performs best.

Feature mimic Inspired by [HML], we construct a feature mimicking baseline (Fig. 7.1c) using our side tasks. Instead of mimicking the embeddings of a squeeze network trained to drive (SAM), we now mimic the embeddings of networks trained for semantic segmentation and stop intentions estimation. Similar to multi-task learning, feature mimicking also imbues the learned features with driving-relevant context but suffers from learning side task-relevant but driving-irrelevant nuisances.

No mimicking We also compare against two baselines that do not use side tasks, SAM-NM and Res101-NM, to analyze the impact of effectively leveraging the side tasks via our method. For both models, we simply use ℓ_1 losses for estimating low-level controls without enforcing the mimicking losses. SAM-NM uses the same two-ResNet architecture as SAM. As using two separate branches to process the image may be suboptimal in the no-mimicking case, we also compare to Res101-NM, a single-ResNet baseline that has a comparable number of network parameters.

From Tab. 7.4, we see that our method outperforms the alternative approaches leveraging

the side task. It outperforms multi-task and feature mimic, showing that it effectively learns a representation that contains only the driving-relevant information associated with the side task and not the nuisances. In addition, it outperforms two-stage-(F), avoiding the perception errors that plague two-stage methods. Finally, we note that all methods using the side tasks outperform the no-mimicking baselines, showing that our choice of side tasks (segmentation masks, stop intentions) is effective for improving generalization and providing driving-relevant context.

7.5.4 Ablation Studies about the Chosen Side Task Annotations

We now conduct a careful ablation study to understand the impact of the chosen individual types of annotations. We analyze the influence of only utilizing one type of knowledge for mimicking: segmentation masks or stop intentions. We have also conducted ablation studies on the importance of each individual category of stop intention values, vehicle, pedestrian and traffic light, and found that the best performance is achieved when all three categories of intentions are used.

We use two different types of knowledge from the squeeze network for mimicking. Segmentation masks provide the mimic network with some simple concepts of object identities and therefore help the agent to learn basic driving skills like lane following and making turns. Meanwhile, stop intentions inform the agent of different hazardous traffic situations that require braking such as getting close to pedestrians, vehicles, or red lights. In Tab. 7.5 we conduct ablation studies mimicking each type of information separately. Both types of knowledge separately bring performance gains, but the best results are achieved only when they are used jointly due to their complementary nature.

	Training weather			New weather			Mean
Mimicking source	Empty	Regular	Dense	Empty	Regular	Dense	
Only stop intention	86 ± 2	47 ± 3	8 ± 4	73 ± 3	53 ± 6	9 ± 5	46.0 ± 1.7
Only seg. mask	**93 ± 1**	50 ± 4	8 ± 4	**85 ± 1**	52 ± 7	7 ± 3	49.2 ± 1.6
Both	92 ± 1	**74 ± 2**	**29 ± 3**	83 ± 1	**68 ± 7**	**29 ± 2**	**62.5 ± 1.4**

Table 7.5: Comparison of mimicking different types of knowledge. We show navigation success rate in the new town on NoCrash. *Only stop intention* and *only segmentation mask* both improve upon the SAM-NM no-mimicking baseline, 31.7%, in Tab. 7.4. The best results are achieved by mimicking both types of embedding knowledge jointly.

7.6 Discussion

We propose squeeze-and-mimic networks, a method that encourages the driving model to learn only driving-relevant context associated with a side task while discarding nuisances. We accomplish this by first squeezing the complementary side task annotations, semantic segmentation and stop intentions, using a squeeze network trained to drive. We then mimic this network's representation to train the mimic network. Our method achieves state-of-the-art on various CARLA simulator benchmarks, including our newly proposed Traffic-school, which fixes previous benchmarks' flaws. In particular, *SAM* outperforms other approaches that also use side tasks.

However, our approach is not without limitations. Though it handles turning commands, it currently does not handle situations requiring negotiating with other agents such as lane changing and high-way merging, a potential topic for future work. Sensor fusion with LiDAR to further improve dense traffic performance could be another interesting direction.

CHAPTER 8

Future Work

With the rapid growth of the autonomous driving industry (Waymo, Cruise, Nuro, *etc.*) and the development of machine learning and deep learning techniques (transformers [VSP17], contrastive learning [CKN20], neural radiance fields [MST20], *etc.*), deep 3d embodied visual recognition is becoming an increasingly more important research area. There are many promising research directions under the wide scope of 3D object and scene understandings and autonomous agent policy learning. We only show a glimpse into the future by discussing some of the valuable problems from different angles.

3D Shape Representations. The evolution of 3D object and scene representations has gone a long way from simple 3D voxels and skeleton graphs to deep implicit surface functions. The pursue of efficient and expressive 3D shape representations is still an ongoing mission. Some important problems to be considered include 1) multi-granularity implicit function designs for hierarchical and efficient representation/reconstruction of 3D shapes; 2) articulated parts modeling with semantic-aware implicit surface functions; 3) curriculum training for fast/robust learning of the dense occupancy or signed distance fields, and so on.

Multi-Modality Fusion in Time. Multi-modality data fusion in time (*e.g.*, sequential point clouds and color images) is critical for the development of 3D object detection and autonomous agent systems. Most of the prior works on 3D detection either are based on a single timestamp or adopt naive early/late fusion schemes for multi-modality fusion, which only have marginal improvements over the LiDAR only baselines [SWC21, YZK21]. With the recent advance of various attention mechanisms and transformer-based fusion architectures

[DCX21, PCG21], multi-modality fusion across time embraces new possibilities.

Self-Supervised Learning on 3D Data. The main-stream methods can be classified into two categories. 1) View prediction-based methods. DeepVoxels [STH19a] and Nerf [MST20] showed that we can learn deep 3D representations directly from multi-view images and the learned representations can be used to generate new data samples. As the next step, it would be interesting to study how can we extend these neural radiance/voxel-based methods to the modeling of dynamic objects and scenes. It is also critical to study what can we do with the learned deep 3D representations (besides view prediction), such as self-driving agent control estimation [HLL19]. 2) Contrastive learning on large-scale point clouds is also a promising research direction. Contrastive learning has been widely applied on 2D images to learn an image-based global representation from un-annotated in the wild images [HFW20]. But for 3D data, such as LiDAR point clouds, we care more about point-wise deep features that carry both local geometry information and awareness of wide receptive field context. Therefore it poses a new requirement of point-wise contrastive learning in order to unsupervisedly extract meaningful features from point clouds [XGG20].

3D Transformer-based Backbones. Transformers demonstrated good context modeling abilities for natural languages and 2D images leveraging self/cross-attention mechanisms [DCL18, DBK20]. For 3D data understanding at scale (*e.g.*, point clouds), besides using PointNet and Graph Neural Network [QSM17a, SGT09], transformers open new research opportunities to design backbones that can effectively and efficiently learn from large-scale point cloud data. Some recently published works include Point Transformer [ZJJ21], Voxel Transformer [MXN21] and so on. The design of the transformer-based 3D data backbones is also tightly connected with the study of self-supervised representation learning strategies. Because we know that the modeling capacities of attention-based transformer models can be most unveiled by large-scale training on unannotated data.

REFERENCES

[AB99] Nina Amenta and Marshall Bern. "Surface reconstruction by Voronoi filtering." *Discrete & Computational Geometry*, **22**(4):481–504, 1999.

[ABC03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. "Computing and rendering point set surfaces." *IEEE Transactions on visualization and computer graphics*, **9**(1):3–15, 2003.

[ABC16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: A system for large-scale machine learning." In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[AMB19] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. "Learning to Reconstruct People in Clothing from a Single RGB Camera." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[AMH78] Brian DO Anderson, John B Moore, and RM Hawkes. "Model approximations via prediction error identification." *Automatica*, **14**(6):615–622, 1978.

[AMO] Sameer Agarwal, Keir Mierle, and Others. "Ceres Solver." `http://ceres-solver.org`.

[AMX18] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. "Video Based Reconstruction of 3D People Models." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[APT19a] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. "Tex2Shape: Detailed Full Human Body Geometry from a Single Image." In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, oct 2019.

[APT19b] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. "Tex2Shape: Detailed Full Human Body Geometry from a Single Image." In *IEEE International Conference on Computer Vision*, 2019.

[ASC11] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. "The wave kernel signature: A quantum mechanical approach to shape analysis." In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1626–1633. IEEE, 2011.

[ASK05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. "SCAPE: Shape Completion and Animation of People." In *ACM Transactions on Graphics*, 2005.

[Ast79] KJ Astrom. "Maximum likelihood and prediction error methods." *IFAC Proceedings Volumes*, **12**(8):551–574, 1979.

[BAD17] Sean L. Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J. Pappas. "Probabilistic data association for semantic SLAM." *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1722–1729, 2017.

[BCP20] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. "Label Efficient Visual Abstractions for Autonomous Driving." *arXiv preprint arXiv:2005.10091*, 2020.

[BKL16] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image." In *European Conference on Computer Vision*, 2016.

[BKO19] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst." In *Robotics: Science and Systems*, 2019.

[BSB07] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. "Detailed Human Shape and Pose from Images." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[BSC00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. "Image inpainting." In *ACM SIGGRAPH*, 2000.

[BSF09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. "PatchMatch: A randomized correspondence algorithm for structural image editing." *ACM Trans. Graph.*, **28**(3):24, 2009.

[BST20] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. "Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction." In *European Conference on Computer Vision*, 2020.

[BTD16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort adn Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. "End to End Learning for Self-Driving Cars." *ArXiv*, **abs/1604.07316**, 2016.

[BTT19a] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. "Multi-Garment Net: Learning to Dress 3D People from Images." In *IEEE International Conference on Computer Vision*, 2019.

[BTT19b] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. "Multi-Garment Net: Learning to Dress 3D People from Images." In *IEEE International Conference on Computer Vision*, 2019.

[CAP20] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020.

[CCR17] Florian Chabot, Mohamed Ali Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. "Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1827–1836, 2017.

[CCS15] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. "High-quality streamable free-viewpoint video." *ACM Transactions on Graphics*, **34**(4):1–13, 2015.

[CFG15a] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. "Shapenet: An information-rich 3d model repository." *arXiv preprint arXiv:1512.03012*, 2015.

[CFG15b] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. "ShapeNet: An Information-Rich 3D Model Repository." Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[CGC15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. "Gated feedback recurrent neural networks." In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

[CH90] Jindong Chen and Yijie Han. "Shortest paths on a polyhedron." In *Proceedings of the sixth annual symposium on Computational geometry*, pp. 360–369. ACM, 1990.

[CHL18] Weikai Chen, Xiaoguang Han, Guanbin Li, Chao Chen, Jun Xing, Yajie Zhao, and Hao Li. "Deep RBFNet: Point Cloud Feature Learning using Radial Basis Functions." *arXiv preprint arXiv:1812.04302*, 2018.

[CKN20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[CKZ15] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "3D Object Proposals for Accurate Object Class Detection." In *NIPS*, 2015.

[CKZ16] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "Monocular 3D Object Detection for Autonomous Driving." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2156, 2016.

[CML18] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. "End-to-end driving via conditional imitation learning." In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

[CMW17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3D Object Detection Network for Autonomous Driving." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526–6534, 2017.

[CP05] Frédéric Cazals and Marc Pouget. "Estimating differential quantities using polynomial fitting of osculating jets." *Computer Aided Geometric Design*, **22**(2):121–146, 2005.

[CRU16] Falak Chhaya, N. Dinesh Reddy, Sarthak Upadhyay, Visesh Chari, M. Zeeshan Zia, and K. Madhava Krishna. "Monocular reconstruction of vehicles: Combining SLAM with shape priors." *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5758–5765, 2016.

[CSC15] Christopher Bongsoo Choy, Michael J Stark, Sam Corbett-Davies, and Silvio Savarese. "Enriching object detection with 2D-3D registration and continuous viewpoint estimation." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2512–2520, 2015.

[CSK15] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving." In *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730, December 2015.

[CSL19] Felipe Codevilla, Eder Santana, Antonio M. Lopez, and Adrien Gaidon. "Exploring the Limitations of Behavior Cloning for Autonomous Driving." In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[CVG14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078*, 2014.

[CWW13] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. "Geodesics in heat: A new approach to computing distance based on heat flow." *ACM Transactions on Graphics (TOG)*, **32**(5):152, 2013.

[CZ19a] Zhiqin Chen and Hao Zhang. "Learning implicit fields for generative shape modeling." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

[CZ19b] Zhiqin Chen and Hao Zhang. "Learning Implicit Fields for Generative Shape Modeling." *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[CZK19] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. "Learning by Cheating." In *Conference on Robot Learning*, 2019.

[DBK20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929*, 2020.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[DC91] E.D. Dickmanns and Th. Christians. "Relative 3D-state estimation for autonomous visual guidance of road vehicles." *Robotics and Autonomous Systems*, **7**(2):113 – 123, 1991. Special Issue Intelligent Autonomous Systems.

[DCL18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805*, 2018.

[DCX21] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. "Dynamic Head: Unifying Object Detection Heads with Attentions." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7373–7382, 2021.

[DFS17] Jingming Dong, Xiaohan Fei, and Stefano Soatto. "Visual-Inertial-Semantic Scene Representation for 3D Object Detection." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3567–3577, 2017.

[Dij59] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs." *Numer. Math.*, **1**(1):269–271, December 1959.

[DQN17] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. "Shape completion using 3d-encoder-predictor cnns and shape synthesis." In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017.

[DRC17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An Open Urban Driving Simulator." In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1–16, 2017.

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, **1**:886–893 vol. 1, 2005.

[FBD19] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. "DeepView: View Synthesis with Learned Gradient Descent." In *CVPR*, 2019.

[Flo62] Robert W Floyd. "Algorithm 97: shortest path." *Communications of the ACM*, **5**(6):345, 1962.

[FNP16] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. "DeepStereo: Learning to Predict New Views From the World'S Imagery." In *CVPR*, 2016.

[FP10] Yasutaka Furukawa and Jean Ponce. "Accurate, dense, and robust multiview stereopsis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(8):1362–1376, 2010.

[Fra17] Uwe Franke. *Autonomous Driving*, chapter 2, pp. 24–54. John Wiley Sons, 2017.

[FS18] Xiaohan Fei and Stefano Soatto. "Visual-Inertial Object Detection and Mapping." *CoRR*, **abs/1806.08498**, 2018.

[FSG17] Haoqiang Fan, Hao Su, and Leonidas J Guibas. "A point set generation network for 3d object reconstruction from a single image." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.

[FWS19] Xiaohan Fei, Alex Wong, and Stefano Soatto. "Geo-supervised visual depth prediction." *IEEE Robotics and Automation Letters*, **4**(2):1661–1668, 2019.

[GAB17] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611, 2017.

[GFK18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. "A papier-mâché approach to learning 3d surface generation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018.

[GFM19] Valentin Gabeur, Jean-Sébastien Franco, Xavier Martin, Cordelia Schmid, and Gregory Rogez. "Moulding humans: Non-parametric 3d human shape estimation from single images." In *IEEE International Conference on Computer Vision*, 2019.

[GG07] Gaël Guennebaud and Markus Gross. "Algebraic point set surfaces." In *ACM Transactions on Graphics (TOG)*, volume 26, p. 23. ACM, 2007.

[GGS96] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. "The lumigraph." In *Siggraph*, volume 96, pp. 43–54, 1996.

[GH97] Michael Garland and Paul S Heckbert. "Surface simplification using quadric error metrics." In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[Gib79] James J. Gibson. *The Ecological Approach to Visual Perception.* Psychology Press, 1979.

[GK19] Riza Alp Guler and Iasonas Kokkinos. "HoloPose: Holistic 3D Human Reconstruction In-The-Wild." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[GKO18] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. "PCPNet Learning Local Shape Properties from Raw Point Clouds." In *Computer Graphics Forum*, volume 37, pp. 75–85. Wiley Online Library, 2018.

[GLD19] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. "The relightables: Volumetric performance capture of humans with realistic relighting." *ACM Transactions on Graphics*, **38**(6):1–19, 2019.

[GLS13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset." *International Journal of Robotics Research (IJRR)*, 2013.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite." *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[GSA09] Juergen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. "Motion capture using joint skeleton tracking and surface estimation." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[GVC18] Andrew Gilbert, Marco Volino, John Collomosse, and Adrian Hilton. "Volumetric performance capture from minimal camera viewpoints." In *European Conference on Computer Vision*, 2018.

[HAS17] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. "Casual 3D photography." *ACM Transactions on Graphics (TOG)*, **36**(6):1–15, 2017.

[HCJ20a] T. He, J. Collomosse, H. Jin, and S. Soatto. "DeepVoxels++: Enhancing the Fidelity of Novel View Syntehsis from 3D Voxel Embeddings." In *Proc. of the Asian Conf. on Comp. Vision (ACCV)*, 2020.

[HCJ20b] Tong He, John Collomosse, Hailin Jin, and Stefano Soatto. "Geo-PIFu: Geometry and Pixel Aligned Implicit Functions for Single-view Human Reconstruction." In *Annual Conference on Neural Information Processing Systems*, 2020.

[HDD92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[HFW20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

[HHY19] Tong He, Haibin Huang, Li Yi, Yuqian Zhou, Chihao Wu, Jue Wang, and Stefano Soatto. "GeoNet: Deep Geodesic Networks for Point Cloud Analysis." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[HKA14] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. "Image completion using planar structure guidance." *ACM Transactions on graphics (TOG)*, **33**(4):1–10, 2014.

[HKJ13] Philipp Heise, Sebastian Klose, Brian Jensen, and Alois Knoll. "PM-Huber: PatchMatch with Huber Regularization for Stereo Matching." *2013 IEEE International Conference on Computer Vision*, pp. 2360–2367, 2013.

[HLC18] Zeng Huang, Tianye Li, Weikai Chen, Yajie Zhao, Jun Xing, Chloe LeGendre, Linjie Luo, Chongyang Ma, and Hao Li. "Deep volumetric video from very sparse multi-view performance capture." In *European Conference on Computer Vision*, 2018.

[HLH17] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. "High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference." In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[HLJ15] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. "Matchnet: Unifying feature and metric learning for patch-based matching." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3279–3286, 2015.

[HLL19] Adam W Harley, Shrinidhi K Lakshmikanth, Fangyu Li, Xian Zhou, Hsiao-Yu Fish Tung, and Katerina Fragkiadaki. "Learning from unlabelled videos using contrastive predictive neural 3d mapping." *arXiv preprint arXiv:1906.03764*, 2019.

[HLR12] David A Hirshberg, Matthew Loper, Eric Rachlin, and Michael J Black. "Coregistration: Simultaneous alignment and modeling of articulated 3D shape." In *European conference on computer vision*, pp. 242–255. Springer, 2012.

[HLZ09] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. "Consolidation of unorganized point clouds for surface reconstruction." *ACM transactions on graphics (TOG)*, **28**(5):176, 2009.

[HML] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. "Learning to Steer by Mimicking Features from Heterogeneous Auxiliary Networks." In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[HPT18] Peter Hedman, Julien Philip, Jan-Michael Frahm True Price, George Drettakis, and Gabriel J. Brostow. "Deep Blending for Free-Viewpoint Image-Based Rendering." *ACM Transactions on Graphics*, 2018.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural computation*, 1997.

[HS19] Tong He and Stefano Soatto. "Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8409–8416, 2019.

[HXL20] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. "ARCH: Animatable Reconstruction of Clothed Humans." *arXiv preprint arXiv:2004.04572*, 2020.

[HXS19] Junning Huang, Sirui Xie, Jiankai Sun, Qiurui Ma, Chunxiao Liu, Jianping Shi, Dahua Lin, and Bolei Zhou. "Learning Driving Decisions by Imitating Drivers' Control Behaviors." *arXiv preprint arXiv:1912.00191*, 2019.

[HXS21] Tong He, Yuanlu Xu, Shunsuke Saito, Stefano Soatto, and Tony Tung. "ARCH++: Animation-ready clothed human reconstruction revisited." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11046–11056, 2021.

[HXZ20] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. "DeepCap: Monocular Human Performance Capture Using Weak Supervision." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[IZZ17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. "Image-to-image translation with conditional adversarial networks." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[JAF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." In *European Conference on Computer Vision*, pp. 694–711. Springer, 2016.

[JBA17] Aaron S Jackson, Adrian Bulat, Vasileios Argyriou, and Georgios Tzimiropoulos. "Large pose 3D face reconstruction from a single image via direct volumetric CNN regression." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1031–1039, 2017.

[JKM17] Dinghuang Ji, Junghyun Kwon, Max McFarland, and Silvio Savarese. "Deep View Morphing." In *CVPR*, 2017.

[JMT18a] Aaron S Jackson, Chris Manafas, and Georgios Tzimiropoulos. "3d human body reconstruction from a single image via volumetric regression." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[JMT18b] Aaron S. Jackson, Chris Manafas, and Georgios Tzimiropoulos. "3D Human Body Reconstruction from a Single Image via Volumetric Regression." *European Conference of Computer Vision Workshops*, 2018.

[JNV20] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. "Exemplar Fine-Tuning for 3D Human Pose Fitting Towards In-the-Wild 3D Human Pose Estimation." *arXiv preprint arXiv:2004.03686*, 2020.

[Joh77] Donald B Johnson. "Efficient algorithms for shortest paths in sparse networks." *Journal of the ACM (JACM)*, **24**(1):1–13, 1977.

[Jol11] Ian Jolliffe. "Principal component analysis." In *International encyclopedia of statistical science*, pp. 1094–1096. Springer, 2011.

[JSZ15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. "Spatial Transformer Networks." In *NIPS*, 2015.

[JZH20] Boyi Jiang, Juyong Zhang, Yang Hong, Jinhao Luo, Ligang Liu, and Hujun Bao. "BCNet: Learning Body and Cloth Shape from A Single Image." In *European Conference on Computer Vision*. Springer, 2020.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction." In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[KBJ18a] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. "End-to-end recovery of human shape and pose." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[KBJ18b] Angjoo Kanazawa, Michael J. Black, David W Jacobs, and Jitendra Malik. "End-to-end recovery of human shape and pose." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[Ken84] David G Kendall. "Shape manifolds, procrustean metrics, and complex projective spaces." *Bulletin of the London Mathematical Society*, **16**(2):81–121, 1984.

[KH13a] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction." *ACM Transactions on Graphics (ToG)*, **32**(3):29, 2013.

[KH13b] Michael Kazhdan and Hugues Hoppe. "Screened Poisson Surface Reconstruction." *ACM Transactions on Graphics*, **32**(3), 2013.

[KHM17] Abhishek Kar, Christian Hane, and Jitendra Malik. "Learning a multi-view stereo machine." In *NIPS*, 2017.

[KLT06] Sing Bing Kang, Yin Li, Xin Tong, and Heung-Yeung Shum. "Image-Based Rendering." *Foundations and Trends in Computer Graphics and Vision*, 2006.

[KPB19a] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. "Learning to reconstruct 3D human pose and shape via model-fitting in the loop." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2252–2261, 2019.

[KPB19b] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. "Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop." In *IEEE International Conference on Computer Vision*, 2019.

[KPD19a] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. "Convolutional mesh regression for single-image human shape reconstruction." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4501–4510, 2019.

[KPD19b] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. "Convolutional Mesh Regression for Single-Image Human Shape Reconstruction." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[KTC15] Abhishek Kar, Shubham Tulsiani, João Carreira, and Jitendra Malik. "Category-specific object reconstruction from a single image." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1966–1974, 2015.

[KWR16] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. "Learning-based view synthesis for light field cameras." *ACM Transactions on Graphics (TOG)*, **35**(6):1–10, 2016.

[LAE16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector." In *ECCV*, 2016.

[LC87a] William E. Lorensen and Harvey E. Cline. "Differentiable monte carlo ray tracing through edge sampling." *ACM SIGGRAPH Computer Graphics*, **21**(4):163–169, 1987.

[LC87b] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm." *ACM siggraph computer graphics*, **21**(4):163–169, 1987.

[LCL07] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. "Parameterization-free projection for geometry reconstruction." *ACM Transactions on Graphics (TOG)*, **26**(3):22, 2007.

[LCT18] Zorah Laehner, Daniel Cremers, and Tony Tung. "DeepWrinkles: Accurate and Realistic Clothing Modeling." In *European Conference on Computer Vision*, 2018.

[LH18] Lorenzo Luciano and A Ben Hamza. "Deep learning with geodesic moments for 3D shape classification." *Pattern Recognition Letters*, **105**:182–190, 2018.

[LKL18] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. "Learning efficient point cloud generation for dense 3d object reconstruction." In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[LKR21] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. "Learning high-speed flight in the wild." *Science Robotics*, **6**(59):eabg5810, 2021.

[LMB05] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, , and Beat Flepp. "Off-Road Obstacle Avoidance through End-to-End Learning." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 739–746, 2005.

[LMR15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. "SMPL: A skinned multi-person linear model." *ACM transactions on graphics (TOG)*, **34**(6):1–16, 2015.

[LMS18] Zhihao Li, Toshiyuki Motoyoshi, Kazuma Sasaki, Tetsuya Ogata, and Shigeki Sugano. "Rethinking Self-driving: Multi-task Knowledge for Better Generalization and Accident Explanation Ability." *ArXiv*, **abs/1809.11100**, 2018.

[LRK17] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V Gehler. "Unite the people: Closing the loop between 3d and 2d human representations." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[LSS18] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. "Deep appearance models for face rendering." *ACM Transactions on Graphics*, **37**(4):1–13, 2018.

[LSS19] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. "Neural volumes: Learning dynamic renderable volumes from images." *ACM Transactions on Graphics (TOG)*, **38**(4):65, 2019.

[LWY18] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. "Cirl: Controllable imitative reinforcement learning for vision-based self-driving." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 584–599, 2018.

[LXS20] Ruilong Li, Yuliang Xiu, Shunsuke Saito, Zeng Huang, Kyle Olszewski, and Hao Li. "Monocular Real-Time Volumetric Performance Capture." In *European Conference on Computer Vision*, 2020.

[LZP19] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. "DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing." *arXiv preprint arXiv:1911.13225*, 2019.

[MAF17] Arsalan Mousavian, Dragomir Anguelov, John J Flynn, and Jana Kosecka. "3D Bounding Box Estimation Using Deep Learning and Geometry." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5632–5640, 2017.

[MBR00] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Leonard McMillan, and Steven Gortler. "Image-based visual hulls." In *ACM SIGGRAPH*, 2000.

[MDG18] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. "Driving Policy Transfer via Modularity and Abstraction." In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1–15, 2018.

[MGK19] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. "Neural rerendering in the wild." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6878–6887, 2019.

[MLH16] Majid Masoumi, Chunyuan Li, and A Ben Hamza. "A spectral graph wavelet approach for nonrigid 3D shape retrieval." *Pattern Recognition Letters*, **83**:339–348, 2016.

[MMK12] Masahiro Mori, Karl F MacDorman, and Norri Kageki. "The uncanny valley [from the field]." *IEEE Robotics & Automation Magazine*, **19**(2):98–100, 2012.

[MMP87] Joseph SB Mitchell, David M Mount, and Christos H Papadimitriou. "The discrete geodesic problem." *SIAM Journal on Computing*, **16**(4):647–668, 1987.

[MNT12] Takashi Matsuyama, Shohei Nobuhara, Takeshi Takai, and Tony Tung. *3D Video and Its Applications*. Springer, 2012.

[MON19a] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Learning 3D Reconstruction in Function Space." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[MON19b] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Occupancy networks: Learning 3d reconstruction in function space." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4460–4470, 2019.

[MSK12] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.

[MSK17] J. Krishna Murthy, Sarthak Sharma, and K. Madhava Krishna. "Shape priors for real-time monocular object localization in dynamic environments." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1768–1774, 2017.

[MSO19] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines." *ACM Transactions on Graphics (TOG)*, **38**(4):1–14, 2019.

[MST20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." *arXiv preprint arXiv:2003.08934*, 2020.

[MXN21] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. "Voxel Transformer for 3D Object Detection." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3164–3173, 2021.

[MXS15] Roozbeh Mottaghi, Yu Xiang, and Silvio Savarese. "A coarse-to-fine model for 3D pose estimation and sub-category recognition." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 418–426, 2015.

[NLT19] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. "Hologan: Unsupervised learning of 3d representations from natural images." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7588–7597, 2019.

[NSH19] Ryota Natsume, Shunsuke Saito, Zeng Huang, Weikai Chen, Chongyang Ma, Hao Li, and Shigeo Morishima. "SiCloPe: Silhouette-Based Clothed People." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked Hourglass Networks for Human Pose Estimation." In *ECCV*, 2016.

[OPB20] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta, and Andreas Geiger. "Learning Situational Driving." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11296–11305, 2020.

[ORF16] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. "Holoportation: Virtual 3d teleportation in real-time." In *Annual Symposium on User Interface Software and Technology*, 2016.

[OTW19] Kyle Olszewski, Sergey Tulyakov, Oliver Woodford, Hao Li, and Linjie Luo. "Transformable Bottleneck Networks." *arXiv preprint arXiv:1904.06458*, 2019.

[PBO20] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. "Exploring Data Aggregation in Policy Learning for Vision-based Urban Autonomous Driving." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11763–11773, 2020.

[PCG21] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. "Multi-Modal Fusion Transformer for End-to-End Autonomous Driving." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7077–7087, 2021.

[PCY16] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles." In *IEEE Transactions on Intelligent Vehicles*, volume 1, pp. 33–55, 2016.

[PE15] Vardan Papyan and Michael Elad. "Multi-scale patch-based image restoration." *IEEE Transactions on image processing*, **25**(1):249–261, 2015.

[PFS19a] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[PFS19b] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

[PGC17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic Differentiation in PyTorch." In *NIPS Autodiff Workshop*, 2017.

[PGM19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "PyTorch: An imperative style, high-performance deep learning library." In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[PKD19] Georgios Pavlakos, Nikos Kolotouros, and Kostas Daniilidis. "Texturepose: Supervising human mesh estimation with texture consistency." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 803–812, 2019.

[PKK19] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. "Revealing scenes by inverting structure from motion reconstructions." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 145–154, 2019.

[Pom88] Dean A. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 305–313, 1988.

[PPH17] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J. Black. "ClothCap: seamless 4D clothing capture and retargeting." *ACM Transactions on Graphics*, **36**(4):1—15, 2017.

[PTX21] Mathias Parger, Chengcheng Tang, Yuanlu Xu, Christopher Twigg, Lingling Tao, Yijing Li, Robert Wang, and Markus Steinberger. "UNOC: Understanding Occlusion for Embodied Presence in Virtual Reality." *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[PYY17] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. "Transformation-grounded image generation network for novel 3d view synthesis." In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 3500–3509, 2017.

[PZ17] Eric Penner and Li Zhang. "Soft 3D Reconstruction for View Synthesis." In *ACM Transactions on Graphics*, 2017.

[PZZ18] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. "Learning to estimate 3D human pose and shape from a single color image." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 459–468, 2018.

[QLW18] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. "Frustum PointNets for 3D Object Detection from RGB-D Data." In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[QSM17a] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation." *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, **1**(2):4, 2017.

[QSM17b] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In *CVPR*, 2017.

[QYS17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In *NIPS*, 2017.

[RCL17] Jimmy S. J. Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. "Accurate Single Stage Detector Using Recurrent Rolling Convolution." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 752–760, 2017.

[REN] RENDERPEOPLE. *http://renderpeople.com/*.

[RGB11] Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning." In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pp. 627–635, 2011.

[RHG15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**:1137–1149, 2015.

[RHP17] Zhihang Ren, Tong He, Lingbing Peng, Shuaicheng Liu, Shuyuan Zhu, and Bing Zeng. "Shape recovery of endoscopic videos by shape from shading using mesh regularization." In *International Conference on Image and Graphics*, pp. 204–213. Springer, 2017.

[RS18] Zhile Ren and Erik B Sudderth. "3D Object Detection with Latent Support Surfaces." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 937–946, 2018.

[RWP06] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. "Laplace–Beltrami spectra as 'Shape-DNA'of surfaces and solids." *Computer-Aided Design*, **38**(4):342–366, 2006.

[SBS10] David Silver, J. Andrew Bagnell, and Anthony Stentz. "Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain." *International Journal of Robotics Research*, **29**:1565–1592, 2010.

[SC15] Shiyu Song and Manmohan Krishna Chandraker. "Joint SFM and detection cues for monocular 3D localization in road scenes." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3734–3742, 2015.

[SGT09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model." *IEEE Transactions on Neural Networks*, 2009.

[SHL18] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. "Multi-view to novel view: Synthesizing novel views with self-learned confidence." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 155–171, 2018.

[SHN19a] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. "PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization." In *IEEE International Conference on Computer Vision*, 2019.

[SHN19b] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2304–2314, 2019.

[SLH19] David Smith, Matthew Loper, Xiaochen Hu, Paris Mavroidis, and Javier Romero. "Facsimile: Fast and accurate scans from an image in less than a second." In *IEEE International Conference on Computer Vision*, 2019.

[SMK15] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. "Multi-view convolutional neural networks for 3d shape recognition." In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[SOG09] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. "A concise and provably informative multi-scale signature based on heat diffusion." In *Computer graphics forum*, volume 28, pp. 1383–1392. Wiley Online Library, 2009.

[SQA15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized Experience Replay." *CoRR*, **abs/1511.05952**, 2015.

[SR] AXYZ design S.R.L. *https://secure.axyz-design.com/*.

[SS86] Micha Sharir and Amir Schorr. "On shortest paths in polyhedral spaces." *SIAM Journal on Computing*, **15**(1):193–215, 1986.

[SSG18] Axel Sauer, Nikolay Savinov, and Andreas Geiger. "Conditional Affordance Learning for Driving in Urban Environments." In *CoRL*, 2018.

[SSK05] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. "Fast exact and approximate geodesics on meshes." In *ACM transactions on graphics (TOG)*, volume 24, pp. 553–560. Acm, 2005.

[SSS16] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving." *NeurIPS 2016 Learning, Inference and Control of Multi-Agent Systems Workshop*, 2016.

[SSS20] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. "PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization." *arXiv preprint arXiv:2004.00452*, 2020.

[STB19] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. "Pushing the Boundaries of View Extrapolation With Multiplane Images." In *CVPR*, 2019.

[STH19a] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. "Deepvoxels: Learning persistent 3d feature embeddings." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2437–2446, 2019.

[STH19b] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. "Deepvoxels: Learning persistent 3d feature embeddings video demo." `https://youtu.be/-Vto65Yxt8s?t=228`, June 2019.

[SWC21] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. "RSN: Range Sparse Net for Efficient, Accurate LiDAR 3D Object Detection." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5725–5734, 2021.

[SZW19] Vincent Sitzmann, Michael Zollhofer, and Gordon Wetzstein. "Scene Representation Networks: Continuous 3D-Strcuture-Aware Neural Scene Representations." In *NeurIPS*, 2019.

[TCF19] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. "Learning spatial common sense with geometry-aware recurrent networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2595–2603, 2019.

[TDB15] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Single-view to multi-view: Reconstructing unseen views with a convolutional network." *arXiv preprint arXiv:1511.06702*, **6**, 2015.

[TH12] T. Tieleman and G. Hinton. "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural Networks for Machine Learning, 2012.

[THB03] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. "Learning Non-Rigid 3D Shape from 2D Motion." In *NIPS*, 2003.

[TNM08] Tony Tung, Shohei Nobuhara, and Takashi Matsuyama. "Simultaneous super-resolution and 3D video using graph-cuts." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[TNM09] Tony Tung, Shohei Nobuhara, and Takashi Matsuyama. "Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo." In *IEEE International Conference on Computer Vision*, 2009.

[TWM19] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. "End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances." *arXiv preprint arXiv:1911.10868*, 2019.

[TZ19] Justus Thies, Michael Zollhöfer, , and Matthias Nießner. "Deferred Neural Rendering: Image Synthesis Using Neural Textures." In *ACM Transactions on Graphics*, 2019.

[TZT18] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. "IGNOR: Image-Guided Neural Object Rendering." In *arXiv*, 2018.

[Ull80] Shimon Ullman. "Against Direct Perception." *Behavioral and Brain Sciences*, **3**:373–381, 1980.

[VBM08] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. "Articulated Mesh Animation from Multi-view Silhouettes." *ACM Transactions on Graphics*, **27**(3):1–9, 2008.

[VCR18a] Gul Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. "Bodynet: Volumetric inference of 3d human body shapes." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 20–36, 2018.

[VCR18b] Gül Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. "BodyNet: Volumetric Inference of 3D Human Body Shapes." In *ECCV*, 2018.

[VPB09] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popovic, Szymon Rusinkiewicz, and Wojciech Matusik. "Dynamic Shape Capture using Multi-View Photometric Stereo." In *ACM SIGGRAPH*, 2009.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[WBS04] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image quality assessment: from error visibility to structural similarity." *IEEE Transactions on Image Processing*, **13**:600–612, 2004.

[WGT17] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. "Interpretable transformations with encoder-decoder networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5726–5735, 2017.

[WHS19] Alex Wong, Byung-Woo Hong, and Stefano Soatto. "Bilateral Cyclic Constraint and Adaptive Regularization for Unsupervised Monocular Depth Prediction." *arXiv preprint arXiv:1903.07309*, 2019.

[WLZ18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[WSK15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. "3d shapenets: A deep representation for volumetric shapes." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[WVT12] Chenglei Wu, Kiran Varanasi, and Christian Theobalt. "Full body performance capture under uncontrolled and varying illumination: A shading-based approach." In *European Conference on Computer Vision*, 2012.

[WZL18] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. "Pixel2mesh: Generating 3d mesh models from single rgb images." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67, 2018.

[WZX16] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." In *Advances in Neural Information Processing Systems*, pp. 82–90, 2016.

[XAJ18] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. "Pointfusion: Deep sensor fusion for 3d bounding box estimation." 2018.

[XCJ19] Xiaogang Xu, Ying-Cong Chen, and Jiaya Jia. "View Independent Generative Adversarial Network for Novel View Synthesis." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7791–7800, 2019.

[XCL15] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. "Data-driven 3D Voxel Patterns for object category recognition." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1903–1911, 2015.

[XCL17] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. "Subcategory-Aware Convolutional Neural Networks for Object Proposals and Detection." *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 924–933, 2017.

[XGG20] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding." In *European Conference on Computer Vision*, pp. 574–591. Springer, 2020.

[XGY16] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. "End-to-End Learning of Driving Models from Large-Scale Video Datasets." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, 2016.

[XJS19] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. "Monocular total capture: Posing face, body, and hands in the wild." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[XW09] Shi-Qing Xin and Guo-Jin Wang. "Improving Chen and Han's algorithm on the discrete geodesic problem." *ACM Transactions on Graphics (TOG)*, **28**(4):104, 2009.

[XWC19] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction." In *Advances in Neural Information Processing Systems*, pp. 490–500, 2019.

[XWL21] Yuanlu Xu, Wenguan Wang, Tengyu Liu, Xiaobai Liu, Jianwen Xie, and Song-Chun Zhu. "Monocular 3D Pose Estimation via Pose Grammar and Data Augmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[XZT19] Yuanlu Xu, Song-Chun Zhu, and Tony Tung. "DenseRaC: Joint 3D Pose and Shape Estimation by Dense Render-and-Compare." In *IEEE International Conference on Computer Vision*, 2019.

[YFH16] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. "Estimation of Human Body Shape in Motion with Wide Clothing." In *European Conference on Computer Vision*, 2016.

[YFH18] Jinlong Yang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, and Stefanie Wuhrer. "Analyzing clothing layer deformation statistics of 3d human motions." In *European Conference on Computer Vision*, 2018.

[YHH19] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. "Pointflow: 3d point cloud generation with continuous normalizing flows." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550, 2019.

[YLF18] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. "Pu-net: Point cloud upsampling network." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2018.

[YLY18] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. "Generative image inpainting with contextual attention." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[YPW17] Yurong You, Xinlei Pan, Ziyan Wang, and Cewu Lu. "Virtual to Real Reinforcement Learning for Autonomous Driving." In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[YRY15] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. "Weakly-supervised disentangling with recurrent transformations for 3d view synthesis." In *Advances in Neural Information Processing Systems*, pp. 1099–1107, 2015.

[YS18] Yanchao Yang and Stefano Soatto. "Conditional prior networks for optical flow." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 271–287, 2018.

[YSG17] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation." In *CVPR*, pp. 6584–6592, 2017.

[YTA20] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. "Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting." *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[YWS19] Yanchao Yang, Alex Wong, and Stefano Soatto. "Dense Depth Posterior (DDP) from Single Image and Sparse Range." *arXiv preprint arXiv:1901.10034*, 2019.

[YWW18] Xiaochuan Yin, Henglai Wei, Xiangwei Wang, Qijun Chen, et al. "Novel view synthesis for large-scale scene using adversarial loss." *arXiv preprint arXiv:1802.07064*, 2018.

[YZK21] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. "Center-based 3d object detection and tracking." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11784–11793, 2021.

[ZBX20] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. "Weakly Supervised 3D Human Pose and Shape Reconstruction with Normalizing Flows." In *European Conference on Computer Vision*, 2020.

[ZHL19] Albert Zhao, Tong He, Yitao Liang, Haibin Huang, Guy Van den Broeck, and Stefano Soatto. "Sam: Squeeze-and-mimic networks for conditional visual driving policy learning." *arXiv preprint arXiv:1912.02973*, 2019.

[ZJJ21] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. "Point transformer." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

[ZPB17] Chao Zhang, Sergi Pujades, Michael Black, and Gerard Pons-Moll. "Detailed, accurate, human shape estimation from clothed 3D scan sequences." In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[ZSS11] M. Zeeshan Zia, Michael J Stark, Bernt Schiele, and Konrad Schindler. "Revisiting 3D geometric models for accurate object shape and pose." *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 569–576, 2011.

[ZSS13] M. Zeeshan Zia, Michael J Stark, and Konrad Schindler. "Explicit Occlusion Modeling for 3D Object Class Representations." *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3326–3333, 2013.

[ZSS14a] M. Zeeshan Zia, Michael J Stark, and Konrad Schindler. "Are Cars Just 3D Boxes? Jointly Estimating the 3D Shape of Multiple Objects." *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3678–3685, 2014.

[ZSS14b] M. Zeeshan Zia, Michael J Stark, and Konrad Schindler. "Towards Scene Understanding with Detailed 3D Object Representations." *International Journal of Computer Vision*, **112**:188–203, 2014.

[ZTF18] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. "Stereo Magnification: Learning View Synthesis using Multiplane Images." In *ACM Transactions on Graphics*, 2018.

[ZTS16] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. "View synthesis by appearance flow." In *European conference on computer vision*, pp. 286–301. Springer, 2016.

[ZYL21] Zerong Zheng, Tao Yu, Yebin Liu, and Qionghai Dai. "PaMIR: Parametric Model-Conditioned Implicit Representation for Image-based Human Reconstruction." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[ZYW19a] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. "Deephuman: 3d human reconstruction from a single image." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7739–7749, 2019.

[ZYW19b] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. "DeepHuman: 3D Human Reconstruction from a Single Image." In *IEEE International Conference on Computer Vision*, 2019.